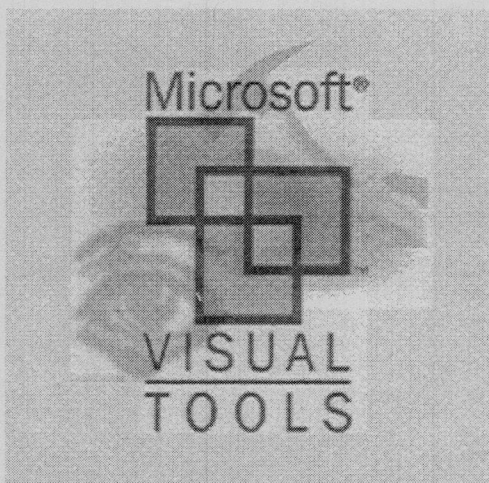


.....



***Microsoft*® Developer
Seminar-in-a-Box Series**

**Microsoft Access 95
Seminar Attendee
Workbook**

Microsoft, Windows, and Win32, are registered trademarks and Access 7.0 is a trademark of Microsoft Corporation. All other trademarks, marked and not marked, are property of their respective owners.

This document is provided for informational purposes only. The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to change in market conditions, it should not be interpreted to be a commitment on the part of Microsoft and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

INFORMATION PROVIDED IN THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND FREEDOM FROM INFRINGEMENT. The user assumes the entire risk as to the accuracy and the use of this document. This document may be copied and distributed subject to the following conditions: 1) All text must be copied without modification (except foreign language translation) and all pages must be included; 2) All copies must contain Microsoft's and Application Developer Training Company's copyright notice and any other notices provided therein; and 3) **This document may not be distributed within the boundaries of Canada or the United States of America.**

Copyright © 1996 Application Developers Training Company. All Rights Reserved.

This seminar is part of the *Microsoft Visual Tools Seminar series*. This material was prepared by Ken Getz and Mike Gunderloy in conjunction with Application Developers Training Company for Microsoft Corporation. All distribution and reprinting is strictly controlled by international copyright laws.

About This Course

Course Schedule

7:00 AM:	Registration
8:00-10:30 AM:	Session 1
10:30-10:45 AM:	Break
10:45-Noon	Session 2
Noon-1:00 PM	Lunch
1:00-2:30 PM	Session 3
2:30-2:45 PM	Break
2:45-4:30 PM	Session 4

Who this Class is For

This class is geared towards developers who are already experienced with Access 2.0 and want to get up to speed quickly using Access 95. But remember, Access makes it easy to develop applications. If you have worked with tables, queries, forms, and reports, there is a lot for you in the new version.

Welcome to Access 95

Microsoft Access for Windows 95, Version 7.0 (referred to as “Access 95” in the rest of the course) is the latest version of Microsoft’s flagship desktop database. This version of Access has been completely rewritten to be more flexible and powerful than ever before. It includes the new version 3.0 of the Jet Engine for faster data retrieval and full 32-bit performance. It also shares the latest version of the Visual Basic language engine with other Office products. You can now write integrated applications that use features from Access, Excel, Project, and other Office-compatible programs.

Like the rest of the Office 95 suite, the Access 95 user interface has been completely redesigned to work as a Windows 95 program. (Access 95 also runs on Windows NT 3.51.) From the new Database Explorer to the extensive use of tabbed dialog boxes, Access 95 looks and feels *right* for Windows 95.

In this course, we will drill down into the new features of Access 95 in three layers:

- ***What’s New for Everyone*** highlights the new user interface, new Wizard capabilities, extensive OLE support, and Office compatibility of Access 95.
- ***What’s New for Developers*** discusses design tool improvements; the new version of the Access Developer’s Toolkit, and replication, which lets you create distributed databases with Access 95; and improvements in OLE support in Access 95.
- ***What’s New for Coders*** looks at the Visual Basic language and the new features it brings to writing code.

You will not see *every* new and improved feature in Access 95. After all, the course is just one day! This course demonstrates the parts of Access 95 that help you work most productively, those that extend the reach of the product to new areas, and those that help you build the best Access 95 applications.

NOTE: We won’t be looking at Access 2.0 today, but don’t worry, your old applications can be converted to run under Access 95 simply by opening them in the new version. You can also run Access 95 and Access 2.0 safely on the same computer (as long as you install them in different folders), so you can work with the old version while you learn the new features. Windows associates your .MDB files with whichever you install last, however, so double-clicking on a database file will start the most recently installed version of Access.

What's New for Everyone?

Managing and Locating Information

This first session is devoted to the new features of Access 95 that make it easier than ever to locate your data and turn it into information. These tools range from new Wizards that help create your database to faster ways to find data in tables and queries. Let's start by looking at improvements to six types of Access objects:

- Databases
- Tables
- Queries
- Forms
- Reports
- Macros

We'll look at Modules this afternoon, in the last part of this course.

Improved Wizards

Access 95 includes two new Wizards to make working with Access 95 easier from the moment you first start building a new application. If you are new to Access, or just tired of reinventing the wheel, use the Database Wizard to create your databases. Use the Answer Wizard to quickly locate help topics, even if you are not sure what information you are looking for.

The Database Wizard

Microsoft spent many hours talking to Access users while they were designing Access 95. One of the things they discovered was that there are common database needs shared by thousands of users. Many people start using Access to track a music collection, or household expenses, or time and billing for a contracting business. The Database Wizard is designed to get these users up and running very quickly. Figure 1 shows the full list of choices that the Database Wizard supplies.

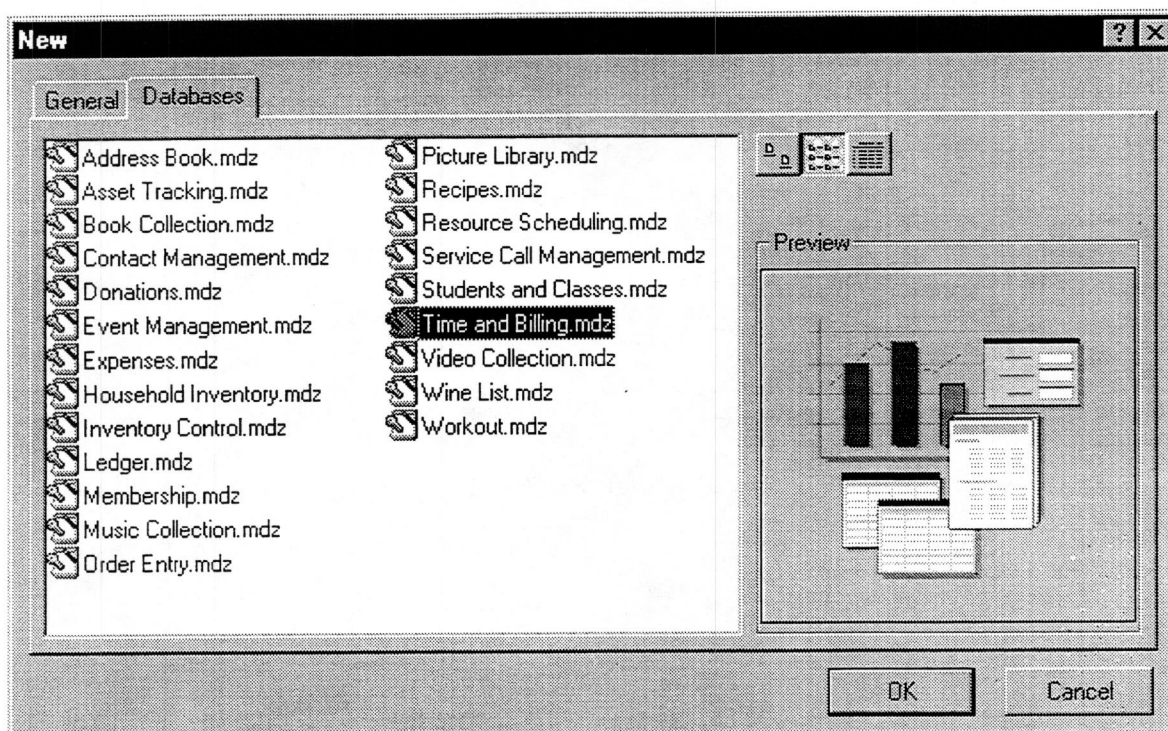


Figure 1. The Database Wizard choices.

When you select a database type from this list, the Database Wizard walks you through all the choices that you need to make to create the database. This includes choosing whether to include optional fields and sample data, choosing a style for forms and reports, and selecting a bitmap for your company logo. The Wizard builds a complete application for you, including

- Tables
- Queries
- Forms
- Reports



Use the Database Wizard to create a Recipes database. Figure 2 shows the Main Switchboard form from this database.

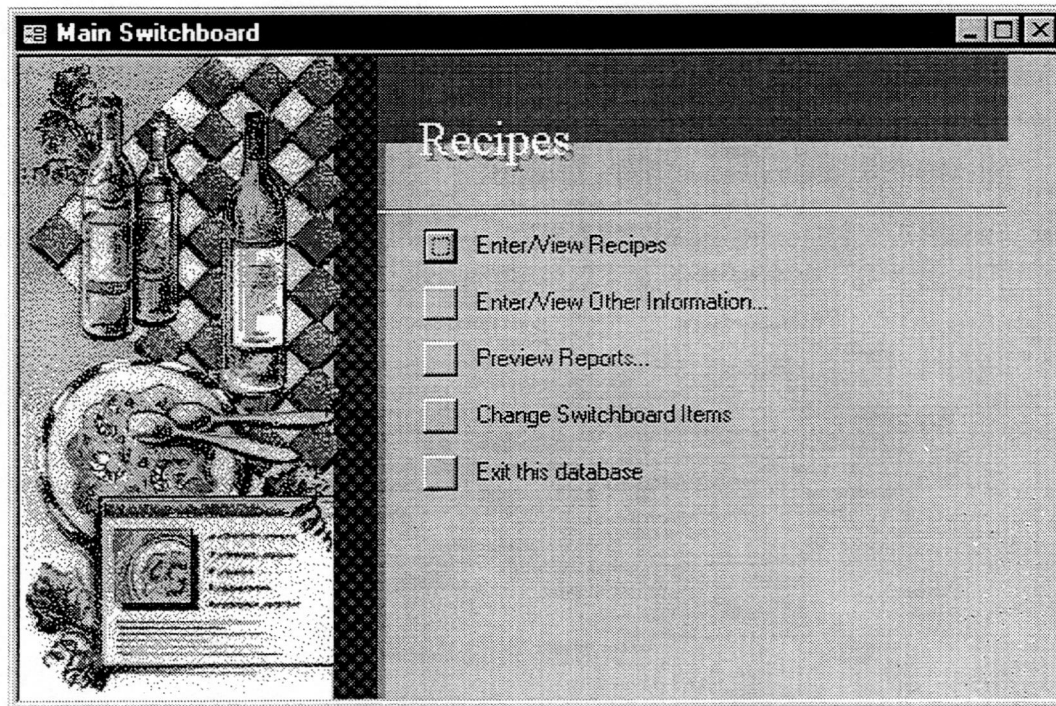


Figure 2. The Main Switchboard form in the Recipes database.

The Answer Wizard

If you have ever had trouble locating information in the Access help file because you did not know what to look for, you will appreciate the Answer Wizard. Rather than forcing you to search for specific terms known only to the Access designers, the Answer Wizard lets you pose your questions in plain English. For example, ask the Answer Wizard "How do I add up sales?" and get the answers shown in Figure 3.

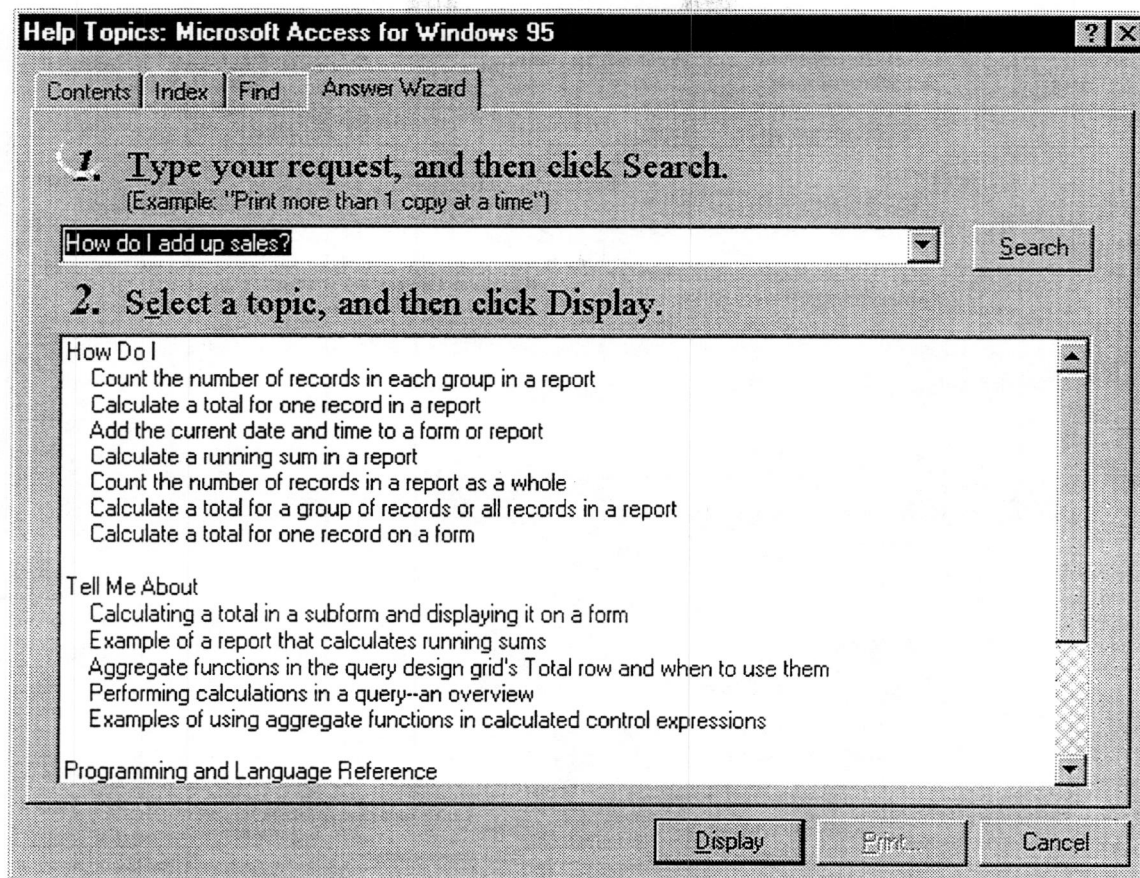


Figure 3. Getting information from the Answer Wizard.

The Answer Wizard has come up with a lot of useful information that relates to the question, even though you did not use “query,” “records,” “total,” “report,” or any of the words that would have been needed to find these topics in a regular help file search.



Ask the Answer Wizard some questions about the sample database.

The Database Explorer

The Database Explorer replaces the old Database Container with a new Windows-95 style interface for managing objects within your Access applications. Figure 4 shows the Database Explorer with the sample database for this course loaded.

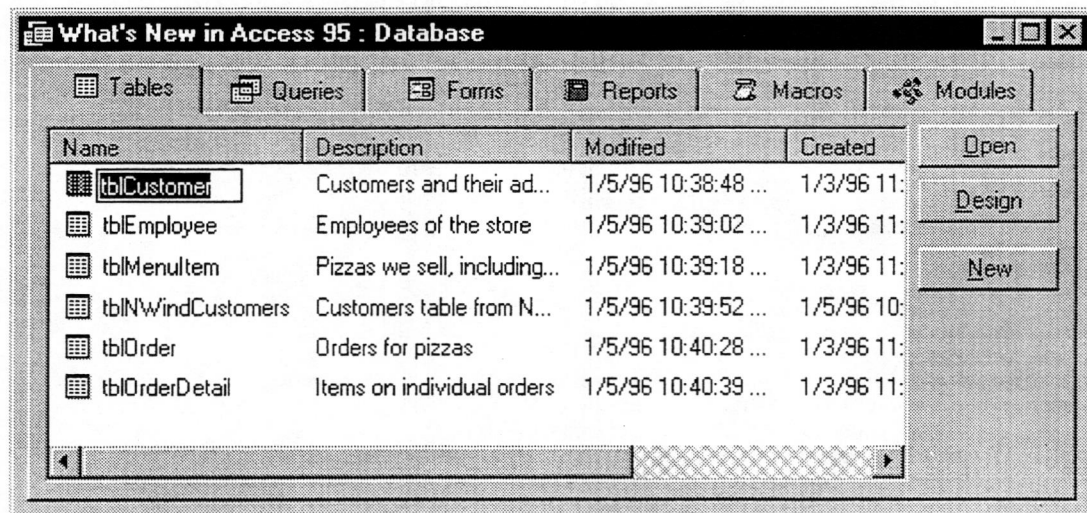


Figure 4. The Database Explorer in Access 95.

The Database Explorer is a very powerful tool. When you work with it, you can:

- View objects in large icon, small icon, list, or details view.
- Sort the objects in details view by clicking on the column headers.
- Click twice slowly on an object to rename it.
- Right-click on an object to get a shortcut menu for that object.
- Right-click on an object and select Properties to view the object's Property Sheet.
- Right-click on an object type tab to get a shortcut menu for the Database Explorer itself.

The Database Explorer also remembers its size and position between uses of the database.



Try some of these features of the Database Explorer in the sample database.

Table Improvements

In Access 95, tables are still the place you store data. What has changed are the tools available to design and edit tables. Power users and developers alike will find things to appreciate in this new toolkit.

Table By Data

Creating simple tables in Access 95 is easier than ever. You no longer have to mess around with table design view. Just click the New Table, select "Datasheet View," and you can begin! You enter your data on a blank datasheet in Table by Data view. As you type the data in, use the arrow keys to move from row to row and column to column. Right-click on the column headers to perform common operations:

- Rename a column.
- Insert a column.
- Delete a column.

When you finish entering your data, just save the table. Access eliminates extra fields and records, assigns data types, and prompts you to add a primary key to the table at this point. Figure 5 shows a table being entered using the Table By Data feature.

	Field1	Field2	Field3	Field4
1	1	1/7/96	2	12.99
2	2	2/1/96	3	22.85
3	1	2/2/96	1	14.01
4	3	3/22/96	4	28.

Record: 4 of 30

Figure 5. Entering data in Table By Data view.

The Lookup Wizard

Access 95 has made it easier than ever to relate tables to one another by introducing the idea of a lookup field. A lookup field is a foreign key that is combined with information about the table containing the primary key to make it easier to build intelligent controls on forms. You look at the new table properties that support lookup fields in a few moments, but first, see how easy it is to create them. You can start the Lookup Wizard by picking "Lookup Wizard" as a data type when you create a new field in a table. As shown in Figure 6, the Lookup Wizard looks (and works) very much like the Combo Box Wizard from Access 2.0.

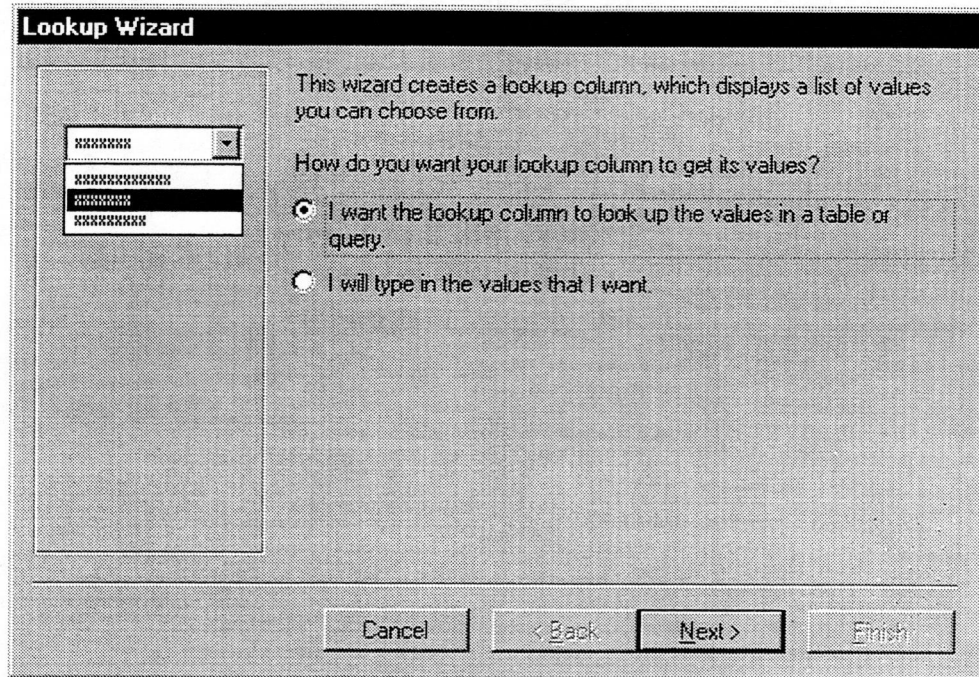


Figure 6. First screen of the Lookup Wizard.

Lookup Properties

The Lookup Wizard works by adding extra properties to the table fields that it creates. Add or edit these properties yourself by switching to the Lookup tab of the field in table design view.

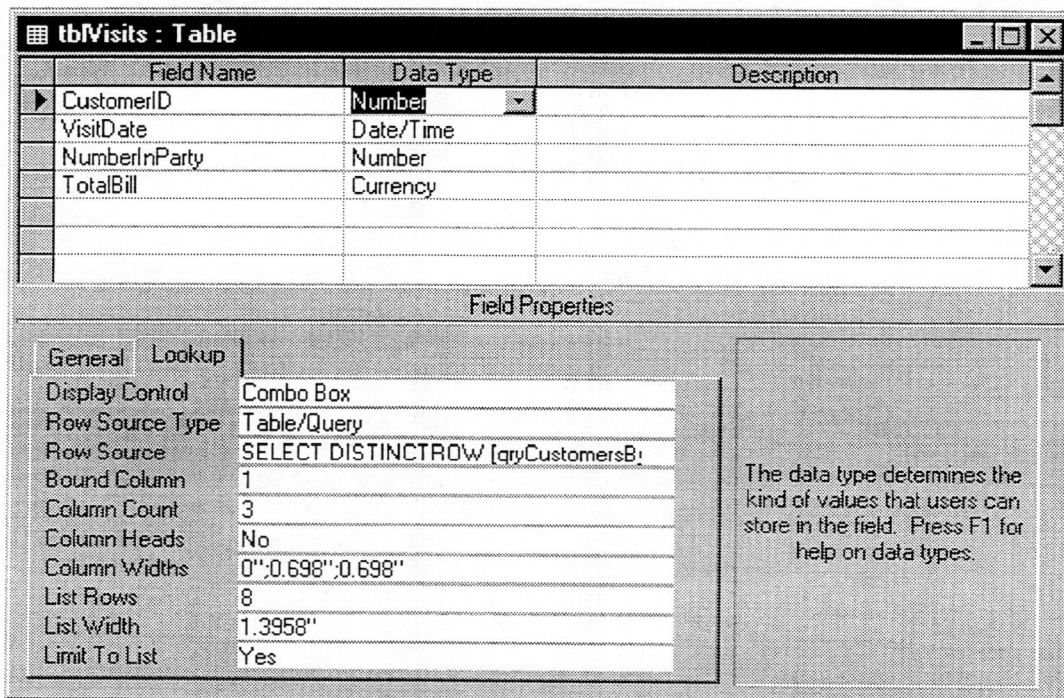


Figure 7. Properties of a lookup field.

Figure 7 shows the properties of a lookup field. Access is very smart about displaying lookup fields. After all, the major reason for setting lookup properties on a foreign key is to save the users of the database the bother of looking at a possibly meaningless primary key. So when you display your table in datasheet view, the lookup field properties are used to create a combo box directly on the datasheet. Figure 8 shows how easy this makes entering data.

CustomerID	VisitDate	NumberInParty	TotalBill
Barks	1/7/96	2	\$12.99
DeLuci	2/1/96	3	\$22.85
	2/2/96	1	\$14.01
Barks	3/22/96	4	\$28.00

Figure 8. Using a lookup field to enter data in a table's datasheet.

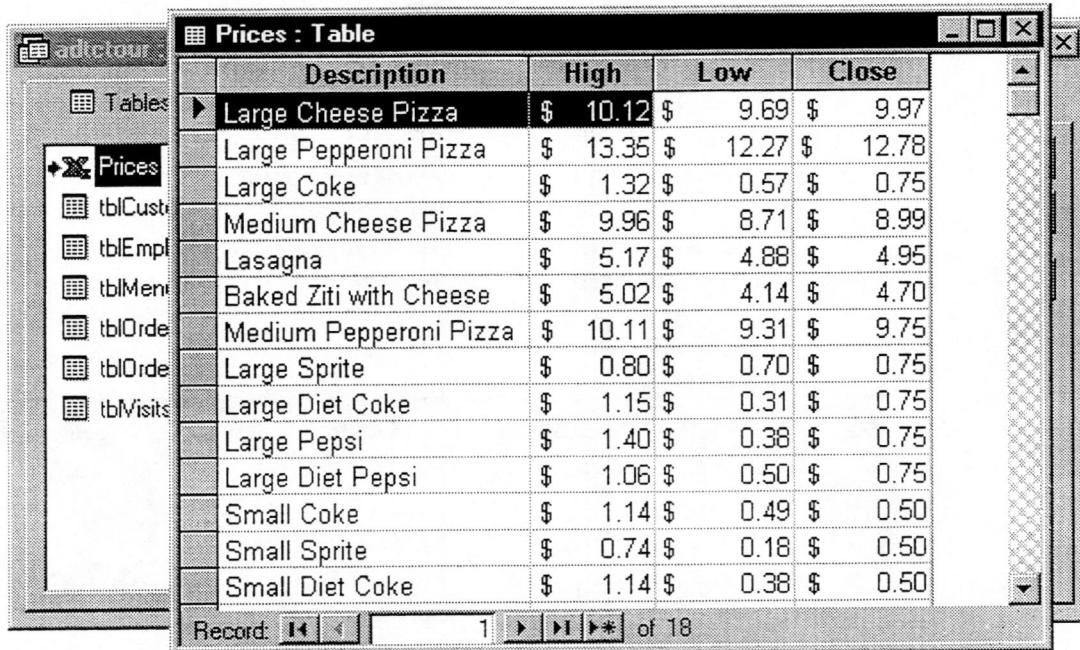
Relationships View

The relationships view is largely unchanged from Access 2.0. However, it has been enhanced with a table "drill-down" feature. While you look at relationships, you can right-click on any table, select Design Table, and see the table in design view.

TIP: You cannot print the relationships view directly from within Access. Instead, use the PrintScr key to capture the window to the clipboard. Then open Paint, paste the captured window in, and print from there.

Link to Excel

What was called *attaching* tables in previous versions of Access is now referred to as *linking* tables. In addition to handling a variety of ISAM databases (FoxPro, dBASE, and Paradox), Access can now use data from Microsoft Excel as a linked table.



Description	High	Low	Close
Large Cheese Pizza	\$ 10.12	\$ 9.69	\$ 9.97
Large Pepperoni Pizza	\$ 13.35	\$ 12.27	\$ 12.78
Large Coke	\$ 1.32	\$ 0.57	\$ 0.75
Medium Cheese Pizza	\$ 9.96	\$ 8.71	\$ 8.99
Lasagna	\$ 5.17	\$ 4.88	\$ 4.95
Baked Ziti with Cheese	\$ 5.02	\$ 4.14	\$ 4.70
Medium Pepperoni Pizza	\$ 10.11	\$ 9.31	\$ 9.75
Large Sprite	\$ 0.80	\$ 0.70	\$ 0.75
Large Diet Coke	\$ 1.15	\$ 0.31	\$ 0.75
Large Pepsi	\$ 1.40	\$ 0.38	\$ 0.75
Large Diet Pepsi	\$ 1.06	\$ 0.50	\$ 0.75
Small Coke	\$ 1.14	\$ 0.49	\$ 0.50
Small Sprite	\$ 0.74	\$ 0.18	\$ 0.50
Small Diet Coke	\$ 1.14	\$ 0.38	\$ 0.50

Figure 9. A linked Excel table in datasheet view.

Figure 9 shows the datasheet view of a linked Excel table. As you can see, it looks just like a normal Access table, and for the most part you can treat it as such.

This capability, combined with the fact that Access is now an OLE Automation Server (you learn about this later in the day), opens up some interesting possibilities for using Access as the user interface for your Excel data. For example, you could link an Excel worksheet to Access, create an Access form to display the data, and then use OLE Automation from Excel to open the Access form whenever you wanted to modify the data. In fact, Excel 95 even includes a DataAccess Form menu item to automate the entire process.

WARNING! One difference between linked Excel tables and other tables is that you cannot delete a row from a linked Excel table. You can, however, delete the contents of every field in the row.



Explore the new table features by adding a few new tables to the sample database and relating them to other tables.

Query Improvements

Queries in past versions of Access were often confusing to create. The new Simple Query Wizard takes much of the drudgery out of creating both Select and Totals queries. Once you create your queries, new formatting and sorting options are available.

Simple Query Wizard

The Simple Query Wizard is a very flexible tool. If you pick a single table and select fields, it builds a simple select query for you and opens it in either design or datasheet view. It can also build queries based on multiple tables. Figure 10 shows this interface in action. After choosing two fields from tblCustomer, you can pick tblOrder and select from its fields without losing the previous selections.

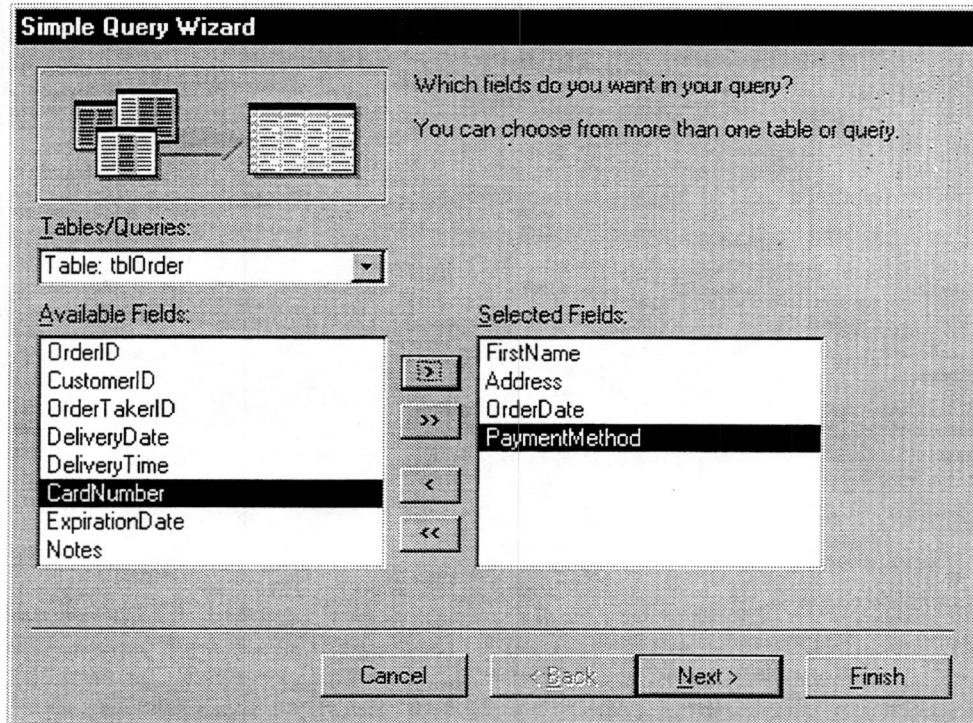


Figure 10. Choosing fields from multiple tables in the Simple Query Wizard.

Even more intelligence is built into the Simple Query Wizard than this example shows. If you select fields from tblCustomer and tblOrderID, the Wizard adds tblOrder, which joins the two tables, to your query. This helps prevent accidentally creating a cross-product query.

TIP: If you find this feature annoying, disable it. Simply clear the "Enable AutoJoin" check box under Tools|Options|Table/Query.

Finally, if you select data for your query that looks like it needs summarizing, the Simple Query Wizard offers to summarize it for you. Figure 11 shows the options the Wizard offers if you create a query including two obviously numeric fields.

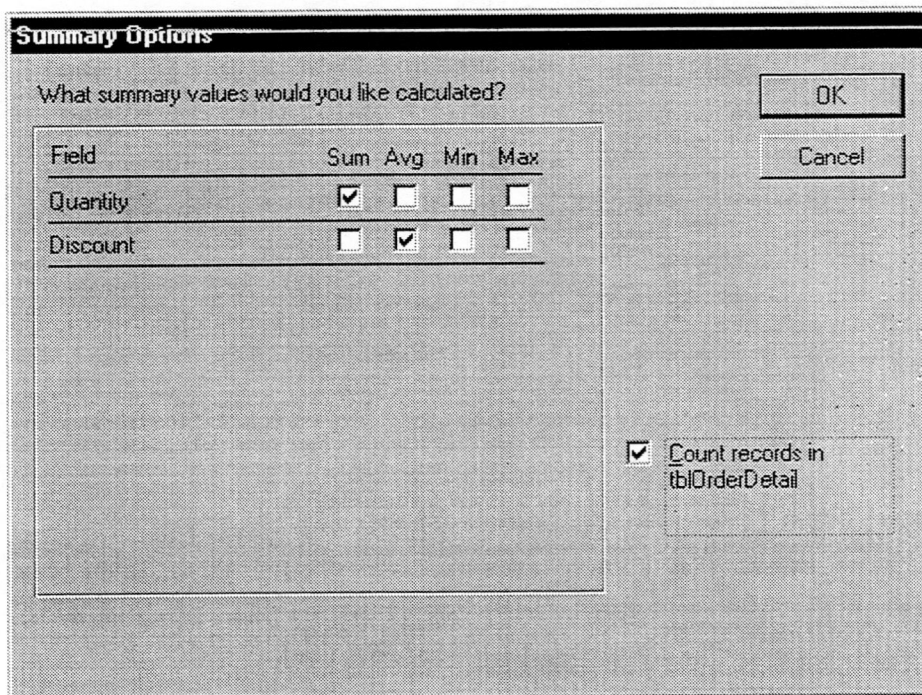


Figure 11. Selecting total types in the Simple Query Wizard.

Datasheet Formatting

You can now save all Access datasheets (not just query datasheets) with formatting. If you customize a datasheet, Access remembers how you wanted it to look and displays it the same way the next time you open it. You also have more choices in how to format your datasheets. You can set:

- Font
- Size
- Bold, italic, underline
- Foreground color
- Background color
- Gridline color

- Cell special effects

Access 95 provides a new Datasheet Formatting toolbar that gives you tools for all of these settings in one place. You can display this toolbar any time you have a datasheet open by just right-clicking on the datasheet toolbar and choosing to display the formatting toolbar.

Access 95 also has a new Cell Effects dialog box, which you can get to with Format/Cells. This dialog box provides a visual interface for setting the look and feel of your datasheets. You can see both the Datasheet Formatting toolbar and the Cell Effects dialog box in Figure 12.

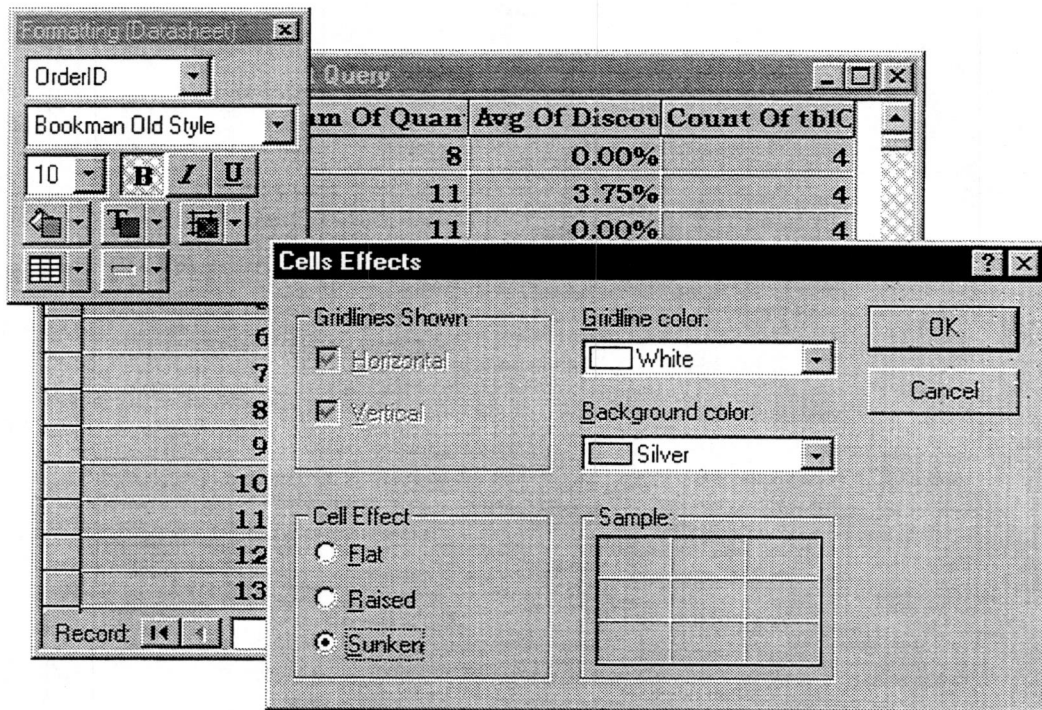


Figure 12. Datasheet formatting tools.

Quicksort Queries

A minor but nice improvement in Access 95 is the inclusion of the Sort Ascending and Sort Descending toolbar buttons for query datasheets.

Demonstrate the Simple Query Wizard.



Form Improvements

Access 95 includes major improvements to the Forms interface with built-in Filter by Selection and Filter by Form capabilities. Before we look at those features, though, let's take a look at the enhancements to the Form Wizards.

Form Wizards

Access 95 includes six different Wizards for creating forms:

- AutoForm: Columnar
- AutoForm: Tabular
- AutoForm: Datasheet
- Form Wizard
- Chart Wizard
- Pivot Table Wizard

The AutoForm Wizards, like their counterpart in Access 2.0, create forms without asking you to make any choices. There are three of them because you can now select from the three different looks shown in Figure 13.

The figure shows three overlapping AutoForm windows for a query named 'qryCustomersByLastName'. The top window is in Columnar mode, the middle in Tabular mode, and the bottom in Datasheet mode.

Columnar AutoForm (Top): Shows individual fields for 'Customer Num1' (value 15), 'Last Name' (value Barks), and a 'Notes' field.

Tabular AutoForm (Middle): Shows a table-like view with columns: Number, Last Name, First N, Address, City, St, Zip C, Phor, Exte, and Notes. It displays two records: one for Barks, Amy (1804 33rd St, Redmon, WA 9811) and one for Boyd, Meg (2001 Avenue, Seattle, WA 981C).

Datasheet AutoForm (Bottom): Shows a standard database table view with columns: CustomerID, LastName, FirstName, and Address. It lists 18 records, with the first few being Barks, Amy; Boyd, Meg; Bustament, Jerry; DeLuci, Phil; Eddy, Jane; Faulcon, Kenneth; Frankel, Carmine; Greg, Gregory; Gunderloy, Mike; Hughes, Paul; Johnson, Phyllis; Jonesy, Mille; and Konick, Gregory.

Figure 13. From top to bottom: Columnar, Tabular, and Datasheet AutoForms.

Microsoft also enhanced the regular Form Wizard in Access 95. Like the Simple Query Wizard, the Form Wizard builds a form based on multiple tables. In fact, it gives you several options when you select multiple tables for a single form. Build a form with a subform, just like the Main-Subform Wizard built in Access 2.0, or build a pair of linked forms with a button on the main form to open the detail form. Figure 14 shows the Form Wizard with these choices.

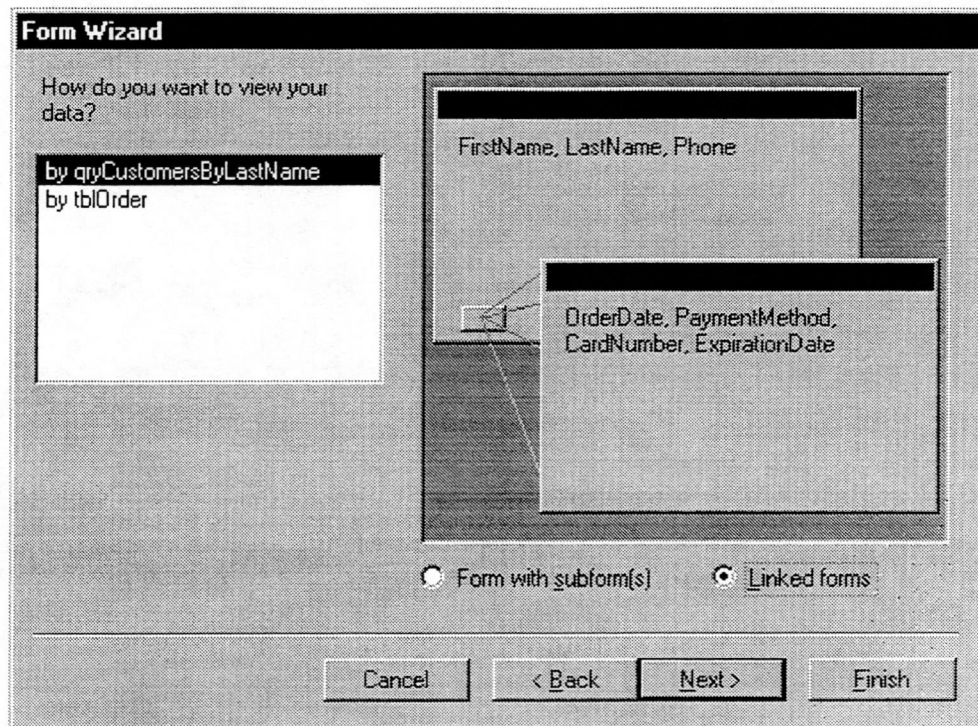


Figure 14. Choosing a method for displaying multiple tables in the Form Wizard.

The Form Wizard also allows you to pick a *style* for your form. A style is a collection of properties for controls, such as font and color, plus a background picture to use on the form. Later in the day you will see how to use the AutoFormat feature to add new styles to this list.

Background Pictures

Of course, you can add a background picture to a form without using the Form Wizard. A collection of new properties for forms controls these pictures. Figure 15 shows these five Picture properties, which let you control the sizing, alignment, and tiling properties of the bitmap, as well as its source and type.

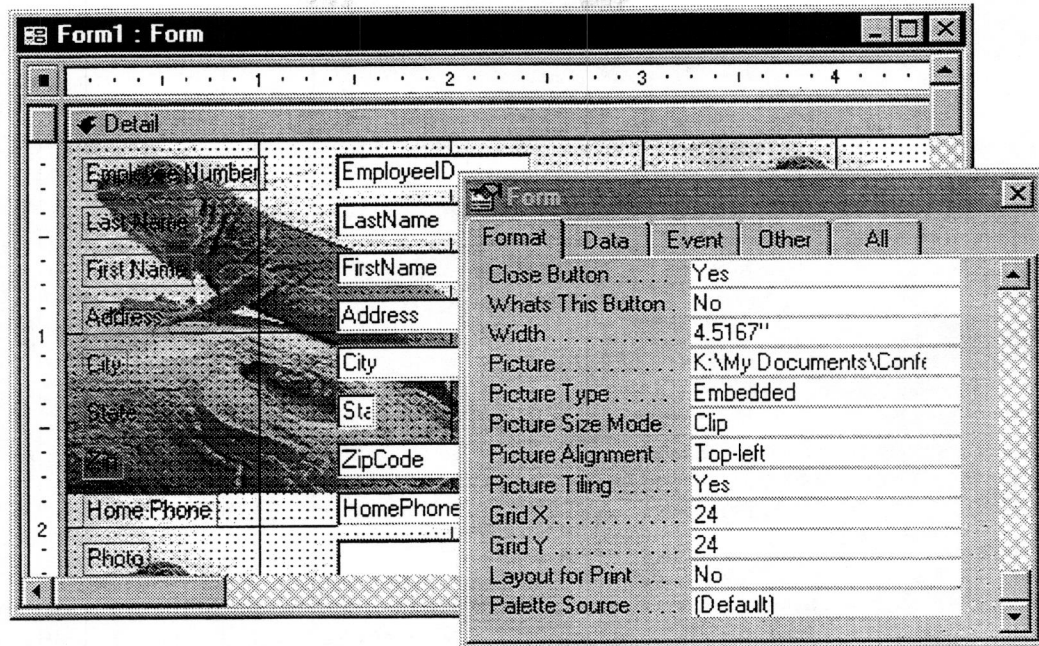


Figure 15. Managing a background bitmap in form design view.

Filter By Selection



Filter by Selection is an extremely simple concept. There are times when you want to find all records in a table that are somehow the same as the filter currently displayed. In Access 95, this is easy. Just place your cursor in the field, click the Filter by Selection button, and only matching records are left on the form. Access even adds the helpful (*Filtered*) marker to your form's navigation bar to warn you that it is only displaying a partial set of the original records.

Filter By Form



Filter by Form provides an alternate and even more flexible way to filter records. When you click the Filter by Form toolbar button, Access constructs a filtering interface from your form itself. This includes combo boxes for the data entry fields on your form, plus a series of tabs for adding additional criteria.

Figure 16. Using the Filter by Form interface to find all customers in Redmond.

Figure 16 shows Filter by Form in action. Opening the City combo box and picking “Redmond” builds a filter to find all the customers who live in Redmond. If you want to find customers in Redmond or Seattle, click the Or tab and select *Seattle* from the same combo. Or to find customers with the last name “Jones” who live in *Redmond*, make two selections on the same page. Once you make all your choices, just click the Apply Filter button on the toolbar, and the real form reappears with the selected records.

TIP: You can also use Filter by Form and Filter by Selection on table or query datasheets.



Build some forms with the Form Wizards and filter the data that they display. Show that even without writing any code or a single macro, Access 95 helps you create complex and powerful ways to view data.

Report Improvements

The major improvement to reports in Access 95 is in creating reports. The new Report Wizard, like the Query Wizard and Form Wizard, has undergone a major revision. Access 95 also supplies a new tool for putting page numbers on reports and a more flexible print preview window than older versions had.

Report Wizards

Access 95 supplies five Report Wizards:

- AutoReport: Columnar
- AutoReport: Tabular
- Chart Wizard
- Label Wizard
- Report Wizard

The AutoReport Wizards, of course, generate quick reports with minimal user intervention. The Chart Wizard and Label Wizard have undergone facelifts, but their functionality is largely unchanged.

The new Report Wizard shares its improvements with the Form Wizard. You can:

- Select multiple tables or queries as the source of the report.
- Select from a variety of ways to organize multiple levels of data.
- Select a layout for the final report.
- Select a visual style for the final report.

Figure 17 shows the new report grouping screen, which helps you visualize the final report as you select grouping levels.

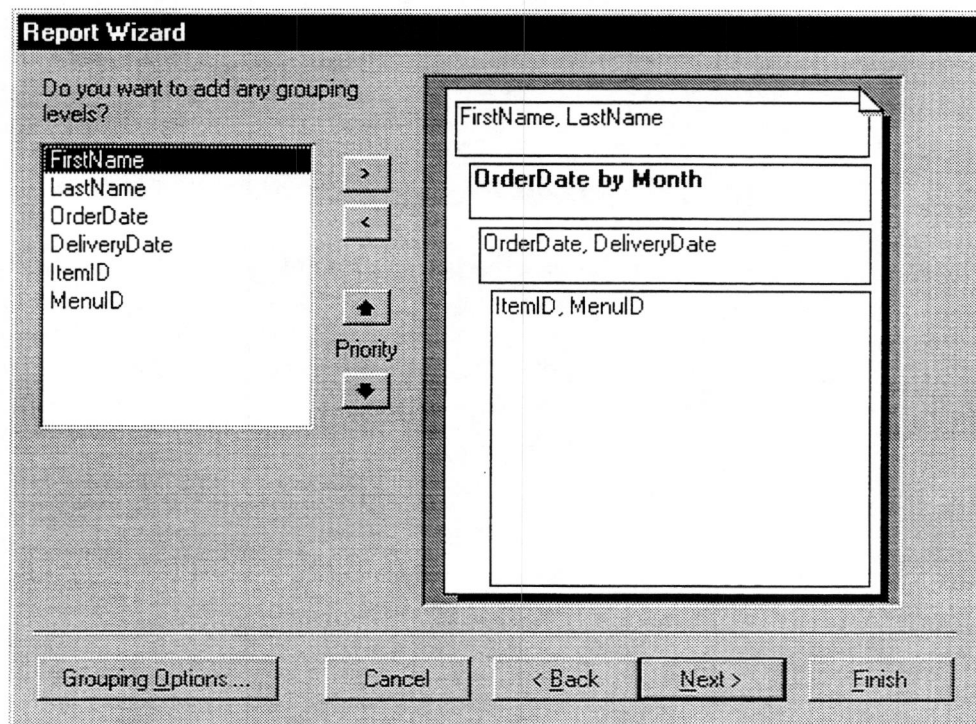


Figure 17. Selecting grouping levels for a report.

Insert Page Numbers

The process of creating page numbers on a report was a frequent source of complaint for Access 2.0 users. In Access 95 it is simple. Just select Insert>Page Number, fill out the Page Numbers dialog box shown in Figure 18, and you are all done.

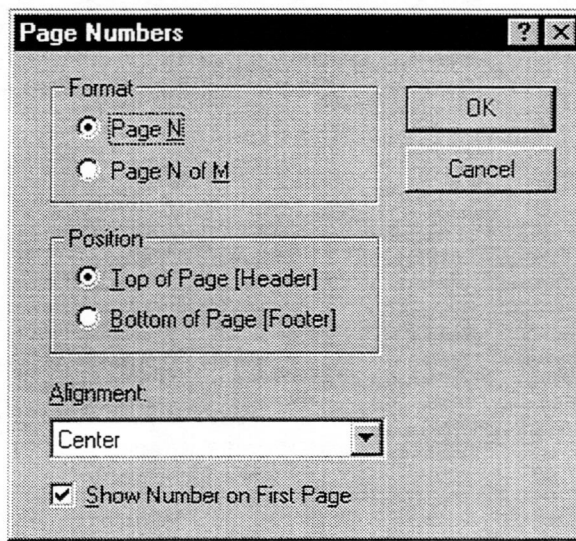


Figure 18. The Page Numbers dialog box.

Print Preview Enhancements

Access 95 now includes both more choices for zoom in print preview and the opportunity to display more pages at once. After opening your report in print preview, right-click anywhere on the preview display and select either *Zoom* or *Pages*. Figure 19 shows three pages of a report at once, in 1x3 view. You can increase this all the way to 4x5 view, to see thumbnails of twenty pages at once. This is useful for getting an overall sense of your report layout.

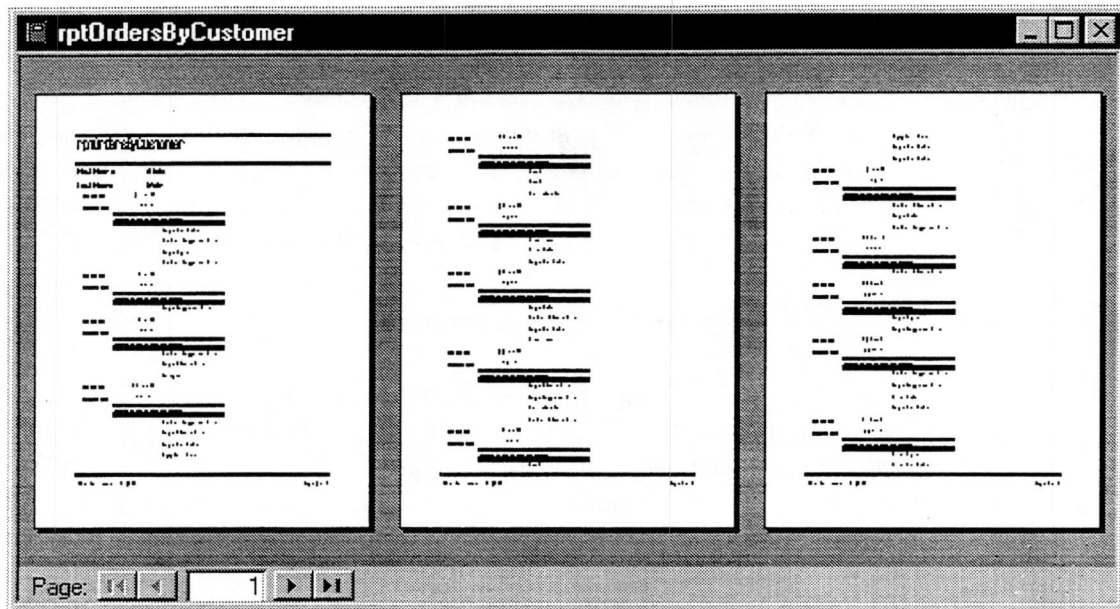


Figure 19. Multipage print preview.

TIP: A major improvement is buried in the Analyze It With MS Excel and Publish It With MS Word features. Access 95 correctly exports reports with subreports to these other formats.



Use the new Report Wizard to quickly build a report in the database, and then show it off with the enhanced print preview features.

Macro Improvements

We'll spend the last session of the course looking at the improvements in the Visual Basic programming language that Access 95 incorporates. But the other programming language, macros, wasn't forgotten in this version either. Access 95 includes several new macro actions, plus a new tool to help you when you switch from macros to Visual Basic. The last session of the course looks at the improvements in the Visual Basic programming language that Access 95 incorporates.

Macro Action Changes

Access 95 includes minor improvements to several existing macro actions (most notably, the Close action now lets you choose whether to automatically save the object you are closing), plus one new macro action. The Save action lets you perform either a Save or a Save As operation on any specified object, or on the object with the focus if you don't specify an object.

Macro to Module Converter

If you have been developing with Access for a while, you may have a substantial investment in macros. As the Northwinds sample database shows, you can carry out very complex operations with macros. When you are ready to start writing Visual Basic code, you can make use of the increased flexibility, error trapping, and debugging tools that this offers.

With Access 95, you can quickly migrate an existing application from macros to Visual Basic. You do this with the new macro to module converter. To use the converter, just select any macro and select File|Save As|Export. Figure 20 shows the new Save As dialog box, which includes *Save as Visual Basic Module*. The rest of the conversion process is entirely automatic.

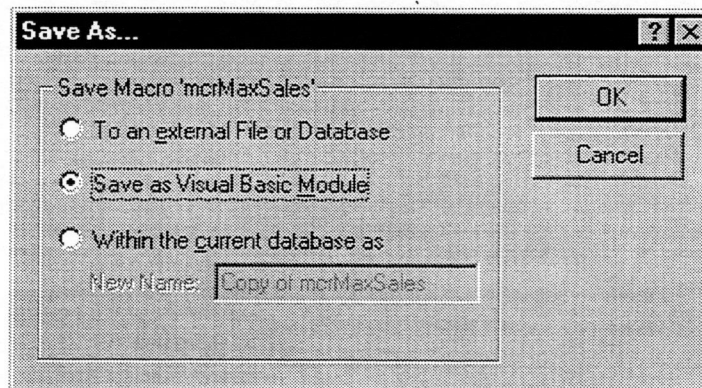


Figure 20. Saving a macro as a Visual Basic module.

WARNING!	Although this process converts any macro to Visual Basic, review the code it creates as your skills increase. While the converter comes up with Visual Basic that performs exactly the operations your macro did, it cannot discover better alternatives for you.
-----------------	---



Convert a fairly complex macro to Visual Basic so you can see this process in action.

Sharing Information

Access 95 is a great tool for sharing information with other people and other applications. Access is a better OLE server and client in this version than ever before. You learn about the programmatic side of OLE this afternoon, but after reviewing Office Compatibility, you look at some of the neat things you can do with it through the user interface without writing any code.

Office Compatibility

Office Compatibility is more than just a marketing buzzword. This is a key piece of the document-centric computing vision that's driving new versions of Office.

Some Office Compatibility features are so pervasive it is not necessary for this course to spend time on them. Things like having the same menu items wherever it makes sense, supporting Long File Names everywhere, and having Tooltips on all controls should be second nature by now for most developers for any Office product.

Improved File Open

Access 95 shares a new File Open dialog box with the other Office 95 products. This dialog box is actually a part of Office, and offers some flexibility not present in the standard common dialog box that most applications use for the same purpose. Figure 21 shows the new dialog box.

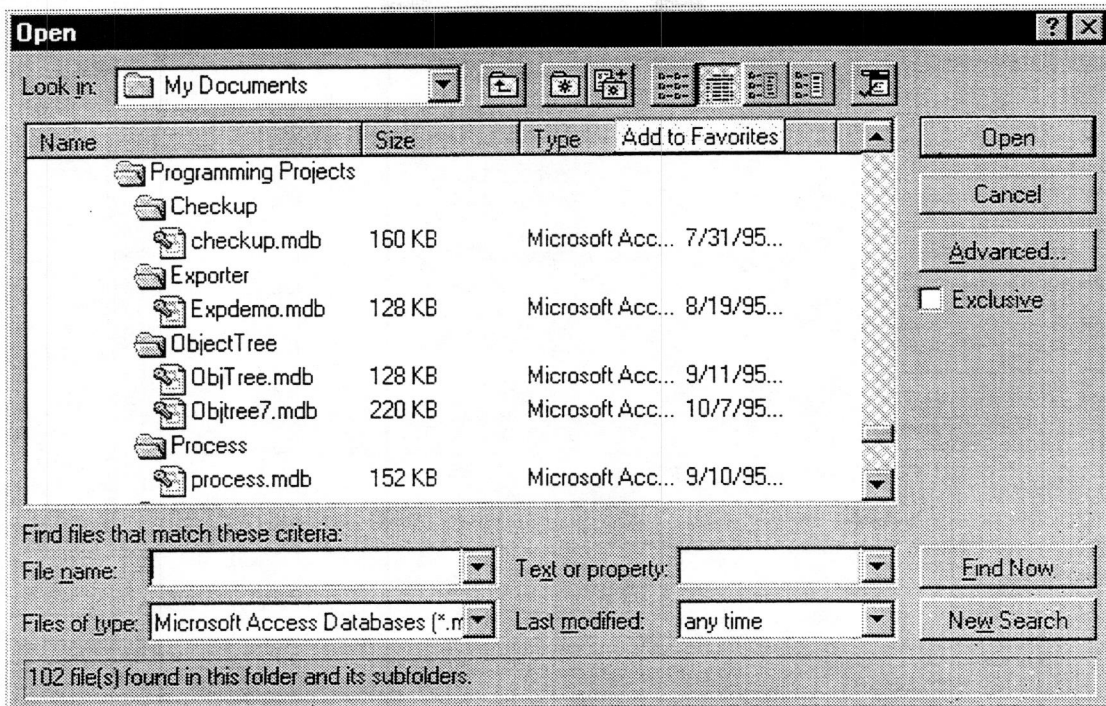


Figure 21. The enhanced File Open dialog box showing subfolders.

One of the most interesting features of this dialog box is its ability to show you files in multiple folders at the same time. To get to the view shown here, navigate to any folder and then select Search SubFolders from the Commands and Settings menu. Of course you can also choose to view files arranged in a list, with details, or a preview (unfortunately, Access 95 does not support putting previews in a database).

The File Open dialog box is also integrated with the File/Database Properties settings within Access itself. For example, suppose you use this feature of Access to add identifying information to a database. Figure 22 shows the Database Properties dialog box for the sample database as it is displayed within Access itself.

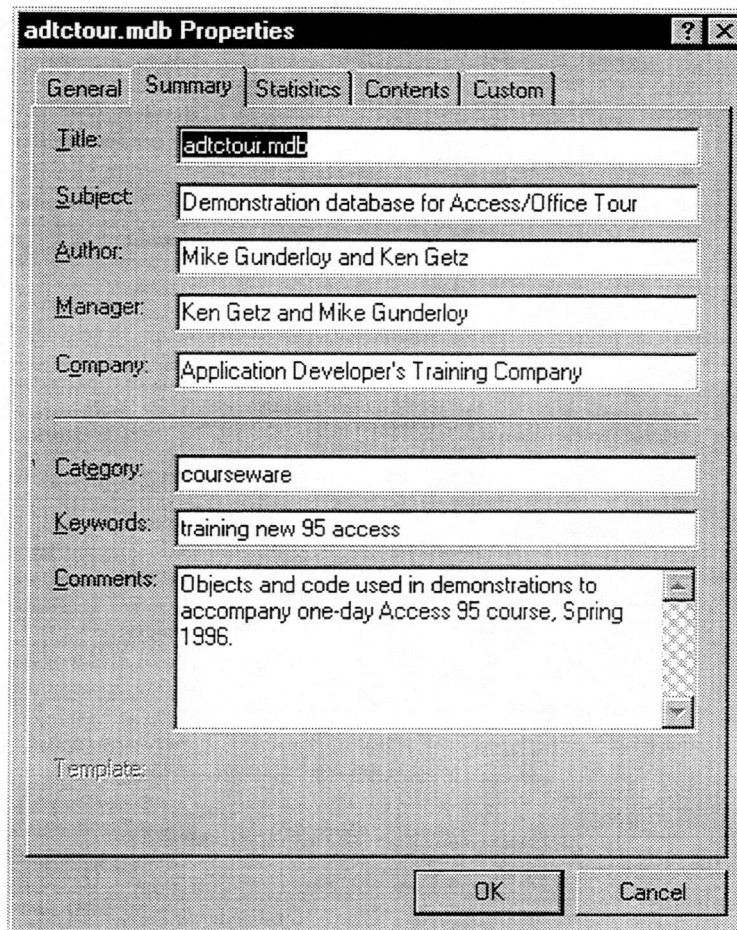


Figure 22. The Summary page of the Database Properties page.

Suppose you know this file is on your hard drive somewhere but you cannot remember which folder it is in or what you called it. Rather than randomly opening files throughout the entire drive, you can use the Advanced button on the File Open dialog box to search for your file by specifying any of these properties. Figure 23 shows the Advanced Find dialog box poised to search for all courseware databases in a particular folder.

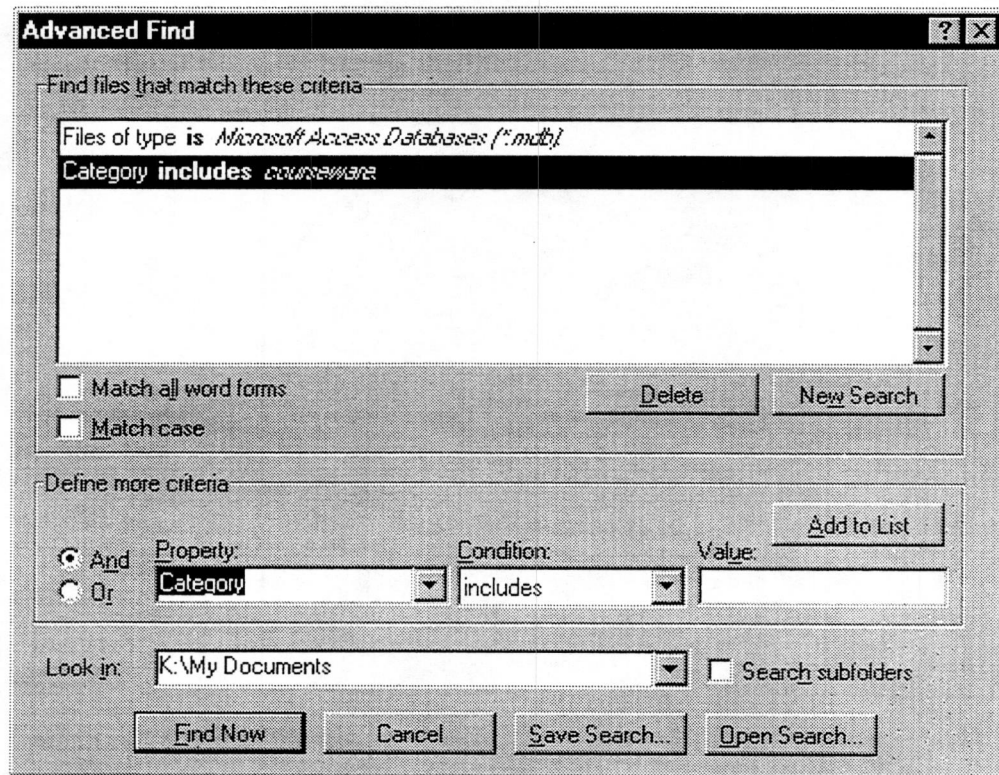


Figure 23. Defining an Advanced Search.

Import Wizard

The Text Import Wizard in Access has been revised to be Office Compatible. In fact, if you know how this tool works in Excel, then you know how it works in Access. One of the nicest enhancements here is that you can create columns visually, by moving column breaks around in a Wizard. Figure 24 shows this column editing in action.

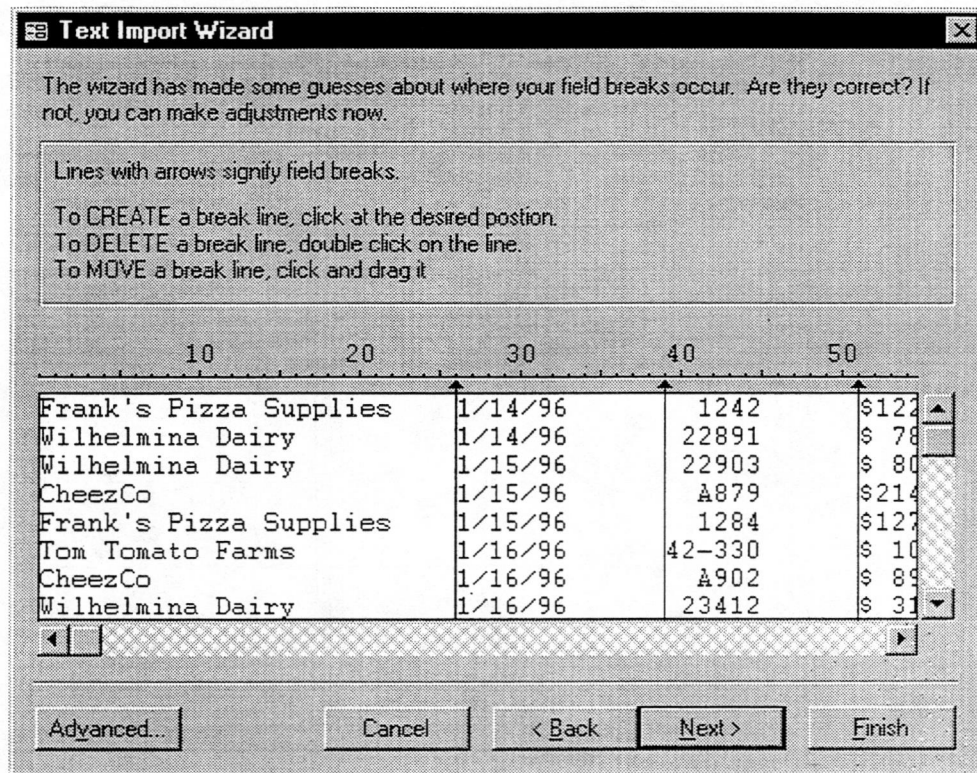


Figure 24. Selecting columns in the Text Import Wizard.



Take a look at some of these Office Compatible features. Find a text file with the new dialog box and import it into the sample database.

Spell Checking and AutoCorrect

Access shares its spell checking tools with the other major Office applications: Word, Excel, and PowerPoint. You can check the spelling in datasheets of all types as well as forms in form view. Access also lets you insert terms into the global AutoCorrect list so that common typos such as *teh* are instantly changed to *the*.

Access does offer one enhancement to the common spelling interface: you can tell the spell checker to ignore a misspelled word only in a single field, rather than in an entire query or table.

Tear-off Palettes

Tear-off palettes were pioneered in Excel. If you format many controls in a single design session, it is a real nuisance to constantly click the toolbar drop-

down buttons to see the color choices. Fortunately, in Office products you do not have to do this. Just drop-down the palette of colors once, use the mouse to grab it in any gray area, and drag it away from the toolbar. It floats on top of your form design surface until you close it. Figure 25 displays the full range of tear-off palettes available in form or report design view.

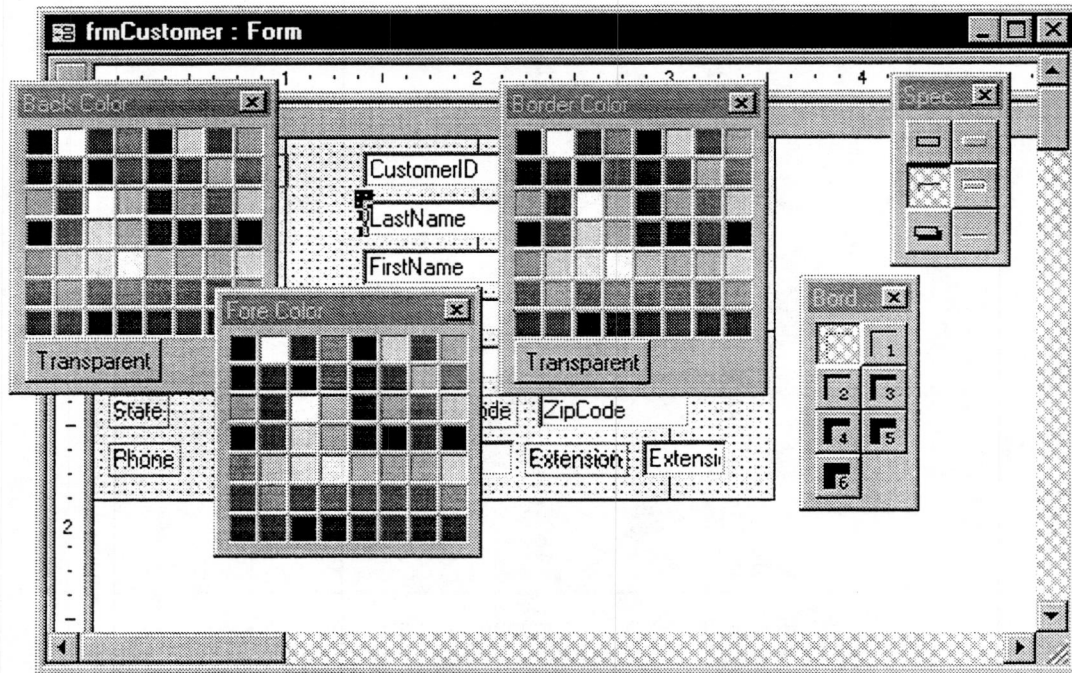


Figure 25. Tear-off palettes in form design view.

TIP: If you have trouble dragging a palette away from the toolbar, double-click in the gray area. This unhooks the palette from the toolbar, so you can drag it wherever you like.

Toolbar Button Editor

Access 95 lets you edit toolbar buttons, just like Word and Excel do. To edit the face of a toolbar button:

1. Right-click on any toolbar.
2. Select Customize....
3. Right-click on any toolbar button.
4. Select Edit Button Image....

This opens the Button Editor with the selected toolbar button loaded. Figure 26 shows the selection of editing tools available to you here. To draw, just pick a color and click where you want a pixel of that color.

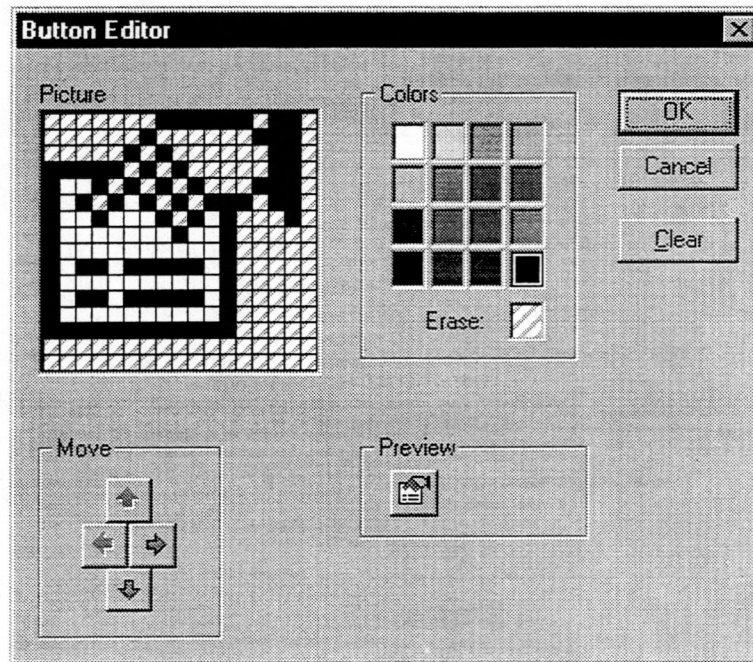


Figure 26. Button editor with a toolbar button loaded.

You can also follow similar steps to take any graphic you have and place it on a toolbar button:

1. Use another application to open the graphic.
2. Copy the graphic to the clipboard.
3. Switch to Access.
4. Right-click on any toolbar.
5. Select Customize....
6. Right-click on any toolbar button.
7. Select Paste Button Image....



Before you look at OLE, take a few moments to test the Office Compatibility features just discussed.

OLE

OLE is a set of Microsoft technologies for interapplication communications. OLE support underlies a diversity of user interface features. In Access 95,

this includes improved drag and drop capabilities, a new Pivot Table Wizard that helps insert Excel objects on your Access forms, Windows 95 shortcut support, and the ability for Excel applications to use Access forms and reports.

OLE Drag and Drop

OLE Drag and Drop is based on a very simple notion: if you see two applications on screen, they ought to be able to see each other. Office applications are very good at working with each other. Generally speaking, you can drag things from one application to another and they arrive in some reasonable format. For example, if you drag a table from Access and drop it:

- On Word, you get a Word table containing the data from the Access table.
- On Excel, you get a worksheet with the table fields and records arranged as columns and records.
- On another instance of Access, you get a copy of the table.

Figure 27 shows what happens when you drop the customer table from the sample database on these three Office applications.

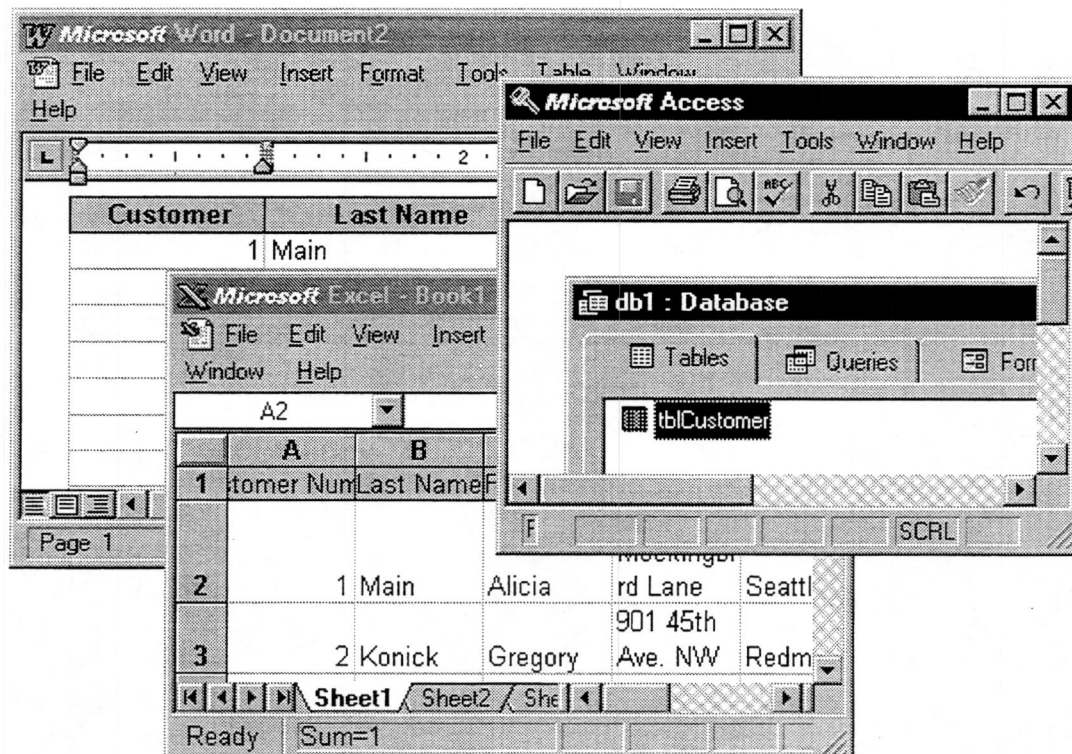


Figure 27. An Access table moved to Word, Excel, and another instance of Access.



Try dragging objects from Access and dropping them on other Office applications.

Pivot Table Wizard

If you are familiar with Excel, you probably know pivot tables. A pivot table is something like an enhanced crosstab query. It is more flexible than an Access crosstab query since it allows you to use three, rather than two, dimensions to analyze your data, and includes a visual interface for rearranging those dimensions.

Access 95 includes a cooperative Wizard that works with Excel. This Pivot Table Wizard creates Access forms with Excel pivot tables as embedded OLE objects. Like any other form wizard, it starts its job in Access, but then it switches to Excel to finish the job. (This is similar to the way the Mail Merge Wizard works together with Word).

The form in Figure 28 shows a Pivot Table form. To change the fields that it is grouped by, just click the button and use the Excel tools to edit it.

OrderDate	Clam chowder	Large Cheese Pizza	Large Coke
5/13/94	0	0	
5/14/94	13	0	
5/15/94	10	4	
5/16/94	1	3	
5/17/94	5	0	
5/18/94	0	5	
5/19/94	0	7	
5/20/94	6	6	
5/21/94	3	13	
5/22/94	0	4	
5/23/94	4	0	
5/24/94	0	0	

Figure 28. A form with an embedded Pivot Table from Excel.



Build another Pivot Table form by starting with the data in qryOrderDetails.

Shortcuts On the Desktop

Access is also integrated with Windows 95 itself. If you shrink the Access workspace so that it does not occupy the entire screen, you can drag objects from the Access database container directly to the Windows 95 desktop. These objects turn into Windows 95 shortcuts that launch the database and open the selected object when you double-click on them. Figure 29 shows a selection of shortcuts on the desktop.



Figure 29. Shortcuts to Access objects.



Create some database object shortcuts and try them out.

What's New for Developers?

Database Tools and Features

Access 95 includes new options and new tools to make it faster to work with and distribute your data. In this session, we'll discuss these features, including:

- Database Options
- The Table Analyzer
- The Performance Analyzer
- Replication

Database Options

The Tools|Options dialog box replaces the old View|Options settings. Figure 30 shows this dialog box, which now uses the Windows 95 tabbed look to organize its information.

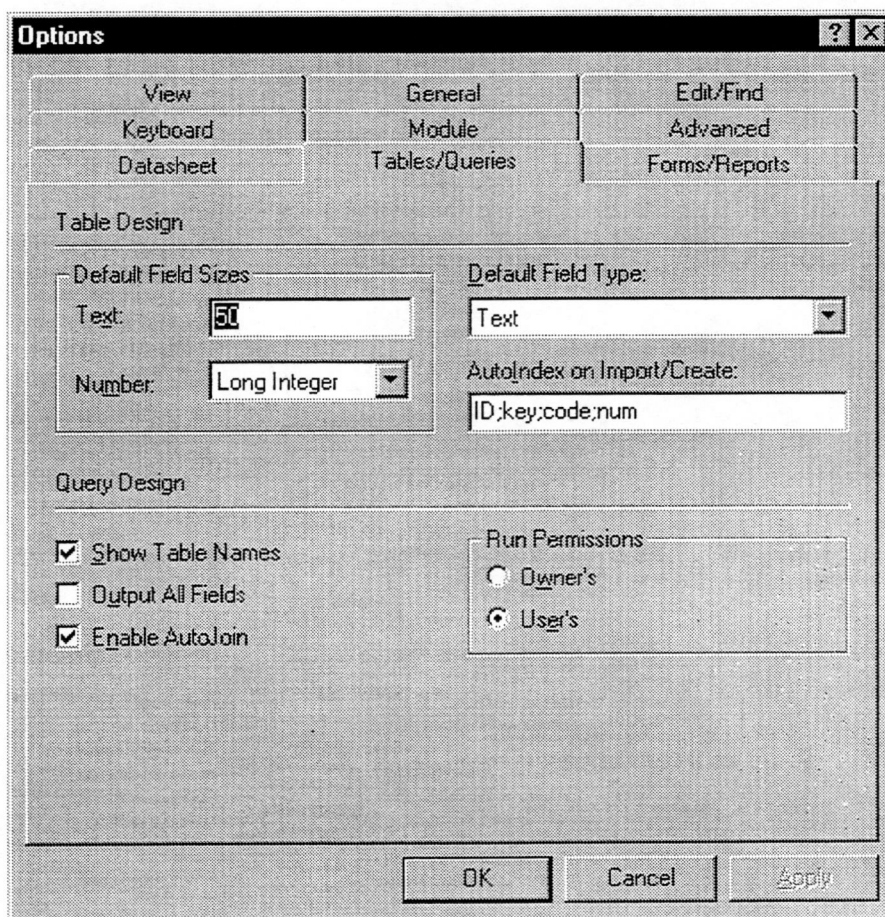


Figure 30. Tables/Queries tab of the Tools|Options dialog box.

In addition to reorganizing options that are familiar to Access 2.0 developers, this dialog box adds some significant new options. These include

- On the Edit/Find tab you can set the maximum number of rows in a combo box in Filter by Form view.
- On the Datasheet tab you can control the default look and feel for datasheets.
- On the Tables/Queries tab you can define AutoIndex keywords that automatically cause fields to be indexed. You can also adjust the default text field width and numeric data type for new fields in table design view.
- On the Module tab you can control the new color-coding and other features of the Visual Basic editor.



Explore these settings in the sample database.

The Table Analyzer

One of the hardest tasks in creating a database is normalizing the data—finding the most efficient structure for your data. There is no “*magic bullet*” to do this for you, since every database is different. But there are rules of normalization that can guide the work of normalizing a database. Access 95 includes a new tool, the Table Analyzer, which can apply some of these rules automatically for you. The Table Analyzer also includes an interactive mode that helps experienced developers quickly break up denormalized tables into normalized parts.

To use the Table Analyzer, select a table in the Database Explorer that contains information on more than one entity. This might be a worksheet of customers and their orders imported from Excel, a table of teachers and students you created without realizing this would require duplicating the teacher information in many records, or a view of accounting data downloaded from a mainframe database server. Then select Tools|Analyze|Table.

The Table Analyzer reviews the data in your table and looks for redundancies that suggest splitting the table into two or more tables. If it finds any, you see a screen like the one in Figure 31. The Table Analyzer also lets you name the new tables, create new Primary Key fields, and create a query that looks like the original denormalized table to your users.

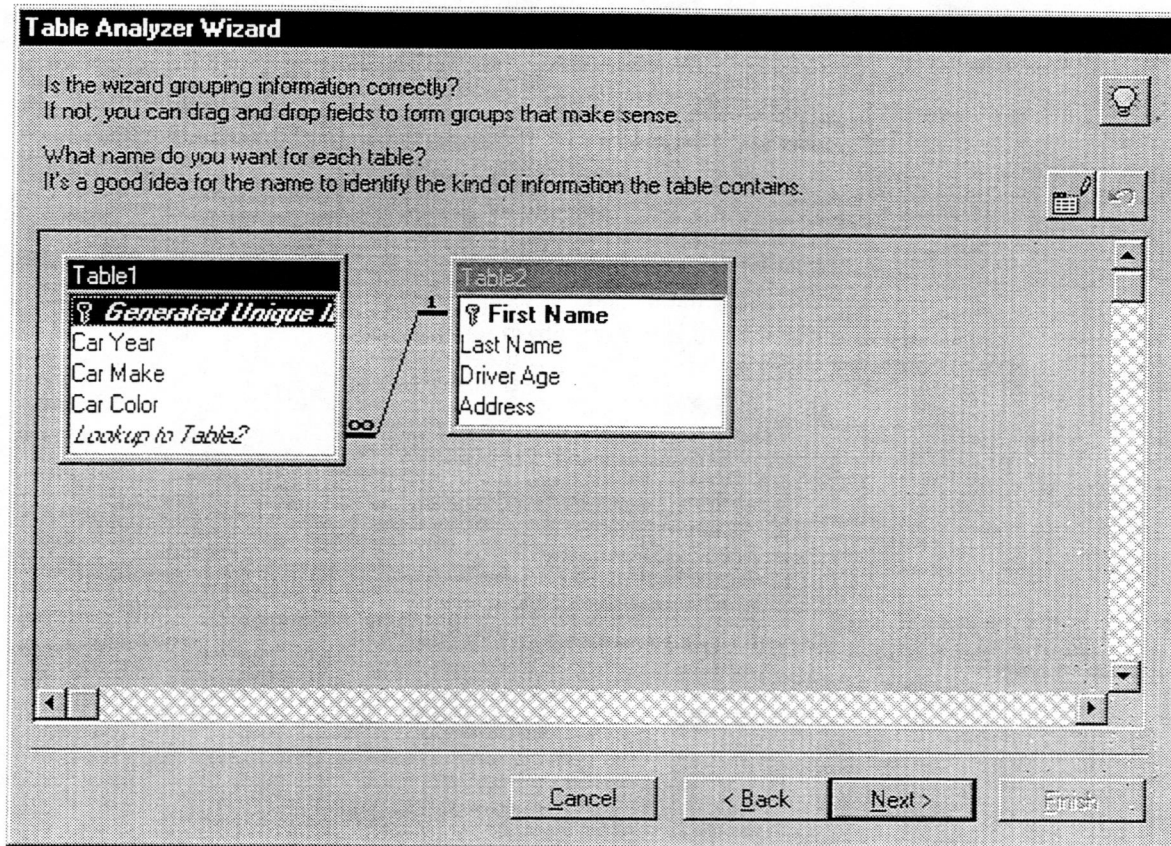


Figure 31. The Table Analyzer at work.

The Table Analyzer also supports manual operation. In this mode, rather than letting the Analyzer decide how to split your data, you decide yourself. You can create new tables just by dragging and dropping fields from an existing table, and the Analyzer takes care of moving all the data around to match your desires.



Use the Table Analyzer to break up a table containing information on drivers and the cars that they own.

The Performance Analyzer

Performance is probably the number one concern of Access developers. There are a lot of things you can do to make a database run faster, but it's not always easy to remember to do them all. The Performance Analyzer is a handy tool that examines an object or group of objects and makes suggestions to improve their performance. For example, it might advise you to add an index to a table

to make a query run faster, or to convert a SQL statement to a saved query to gain the benefits of precompilation.

To run the Performance Analyzer, select Tools|Analyze|Performance and follow the on-screen instructions.



Watch a demonstration of the Performance Analyzer to see what it can do on a database that is not already properly normalized.

Replication

Replication is a new feature of Access 95 and Jet 3.0 that lets you create multiple copies of a database, distribute the copies to desktops throughout your office—or throughout the world, and keep the databases perfectly synchronized. All this with no programming required! (Though you can, of course, control this all programmatically.)

Keeping multiple copies of a database synchronized using Access 2.0 was certainly possible, but required tons of programming and the maintenance of audit logs that recorded every modification made to the database. Now, with Access 95's built-in support for replication, you reap the benefits of distributed computing with very little effort.

Why Would You Want to Use It?

You can benefit from Access replication in many applications. For example, you might use replication for:

- Distributing copies of a database to satellite offices.
- Distributing your sales database to your mobile salespeople.
- Maintaining a warm backup of your shared Access database.
- Balancing the load on an overworked server database.
- Distributing application updates.

How It Works

There are three major parts to replication:

1. Replication

2. Synchronization
3. Conflict resolution

The first step to using the replication services in Access 95 is to *replicate* the database. When you replicate a database, Access makes changes to the database that allows it to track all changes for easy synchronization with other replicated copies of the same database.

When you first replicate a database, Access creates the *design master* of a new *replica set*. From the design master, you create additional members of the replica set, called *replicas*. You can change data in any replica of the replica set, but you can make design changes—for example, changing the layout of a form—only in the design master.

Once you have replicated a database and distributed the replicas, you exchange changes between replicas by *synchronization*. When you synchronize a pair of replicas, Access exchanges all design and data changes between the two replicas.

As part of the synchronization process, Access keeps track of *conflicts* between replicas. A conflict occurs whenever two users make contradictory changes to the same record. Access automatically picks a winner in case of a conflict, then notifies the other user to provide the chance of a manual override. Access also allows you to resolve conflicts programmatically.

Replicating a Database

Select Tools|Replication|Create Replica to convert a regular Access database into a replicated design master. Access automates these steps:

1. Close the database.
2. Create an optional backup.
3. Change the original database to a Design Master.
4. Create a new replica.
5. Re-open the Design Master.

TIP: It is a good idea to let Access make a backup before it turns your database into a replica, because it is difficult to go backwards and undo a replicated database.

To create an additional replica, open any existing replica and select Tools|Replication|Create Replica.



Watch a demonstration to see how to convert a regular Access database into a design master and then create a second replica from the design master.

Synchronizing Replicas

Once you distribute replicas of a replica set, you can make updates to the data in each replica independently of the others. (As previously mentioned, you can only make design changes to the design master.) To exchange updates between two replicas, you must initiate a synchronization exchange.

Select Tools|Replication|Synchronize. Now initiate a synchronization exchange between the currently-open database and another replica. When you do this, Access displays the Synchronize Database dialog box as shown in Figure 32.

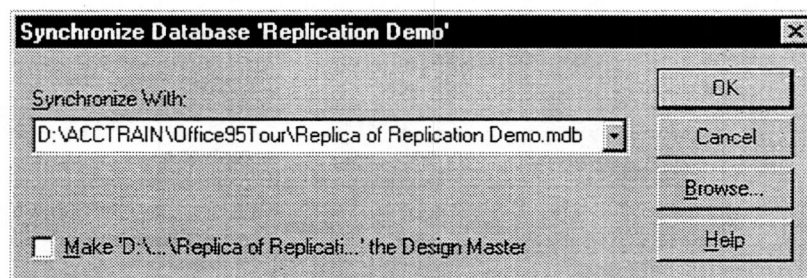


Figure 32. This dialog box appears when you select Tools|Replication|Synchronize Now.

Two factors affect how updates are propagated through the replicas in a replica set:

- **The synchronization topology:** which replicas exchange updates with which others. The most commonly-used topology is the star topology, where one replica—designated the hub replica—initiates all synchronization exchanges with each other member of the replica set. Other topologies are also possible. For more information, consult the *Microsoft Jet Database Replication* whitepaper.
- **The synchronization schedule:** how often you synchronize replicas. The more often you synchronize, the quicker updates are propagated through the replica set. On the other hand, if you synchronize too often, you create excessive network traffic.

Managing Conflicts

When users update the same record in two replicas, Access faces a conflict that it must resolve. The algorithm Access uses to declare the winner of the conflict is simple:

- If one version of the record was saved with updates more times, Access declares it the winner.
- If both versions of the record were saved with updates an equal number of times, Access randomly selects one version to be the winner.

Fortunately, Access provides a mechanism for overriding this automated decision. Access reports all conflicts to the losing replica and informs you of the presence of conflicts the next time the replica is opened. Of course, if there are multiple conflicts, it is likely that both replicas will lose at least one conflict.

The next time you open a database that has lost a conflict, Access displays the *Conflict Resolution Wizard* shown in Figure 33.

Resolve Replication Conflicts - tblCustomer	
Existing Record	Conflict Record
CustomerId: 1	CustomerId: 1
LastName: Johnson	LastName: Johnson
FirstName: Alicia	FirstName: Alicia
Address: 1313 Mockingbird Lane	Address: 1313 Mockingbird Lane
City: Seattle	City: Seattle
State: WA	State: WA
ZipCode: 98118	ZipCode: 98115
Phone: 20688869	Phone: 20688869
Extension:	Extension:
Notes: A great customer!	Notes:
<input type="button" value="Keep Existing Record"/> <input type="button" value="Overwrite with Conflict Record"/>	
Record: 1 of 1	

Figure 33. The Access Conflict Resolution Wizard builds a custom form for each conflict table.



In this demonstration, we'll create a conflict between two replicas, synchronize them, and show how to use the Access Conflict Resolution Wizard to resolve the conflict.

Programmability

All parts of Access 95 replication are completely programmable, including:

- Replicating a database
- Synchronizing replicas
- Resolving conflicts

You can programmatically manipulate replication from Access 95, Visual Basic 4.0, Visual C++ 4.0, and any product that can act as an OLE Automation client (e.g., Excel) using Data Access Objects (DAO).

For example, the following Access function performs a synchronization exchange between the currently-opened database and "*Replica of Replication .Mdb*" replica:

```
Function Synch()  
    Dim dbCurrent As Database  
  
    Set dbCurrent = CurrentDb()  
    dbCurrent.Synchronize _  
        "c:\demo\Replica of Replication Demo.Mdb"  
End Function
```

Other Replication Tools

In addition to using Access and DAO to manage replication, you can also use the Windows 95 Briefcase or Replication Manager.

If all you need is to keep two copies of a database synchronized, you can use the Windows 95 Briefcase to manage things transparently. Just drag the database into the briefcase on one machine, take the briefcase to another machine, and work on the replica there. When you want to synchronize replicas, use Briefcase's Update All command. This form of replication is

ideal for the user who wants to share a database between desktop and laptop, or work and home.

At the other end of the spectrum, Replication Manager is a component of the Access 95 Developer's Toolkit. While Replication Manager duplicates much of the functionality built into Access, it also contains several unique features that make it especially useful for managing replication in a corporate environment. These features include:

- A facility for performing scheduled, unattended synchronizations without any programming.
- Tools for reviewing synchronization logs.
- Tools for managing remote replicas.

The Replication Manager is shown in Figure 34.

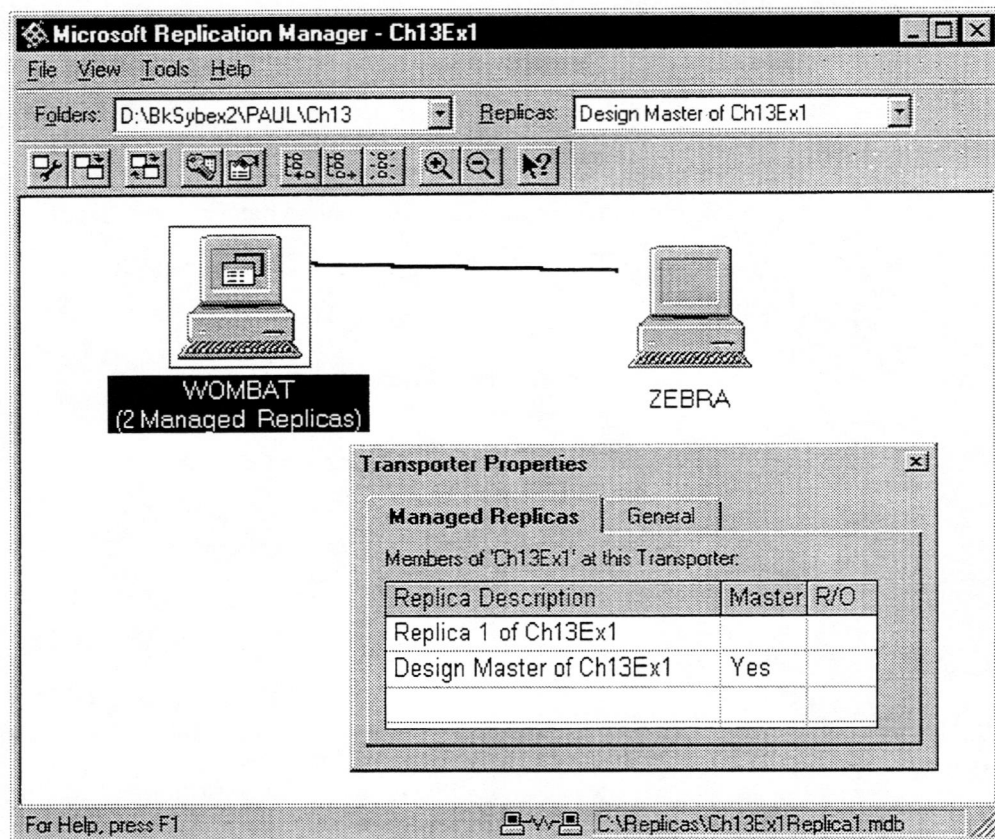


Figure 34. Replication Manager is being used to manage replicas located at two sites: Wombat and Zebra.

Solution Distribution

Access 95 offers new features to make the task of distributing your application to its ultimate users faster and easier. These include the new Startup Properties, and Application Splitter Wizard, and a vastly revised Setup Wizard (included in the Access Developer's Toolkit).

Startup Properties

Access 95 allows you to replace the simple AutoExec macro with a flexible set of startup properties. When you select Tools|Startup, you see the dialog box in Figure 35, with a variety of choices.

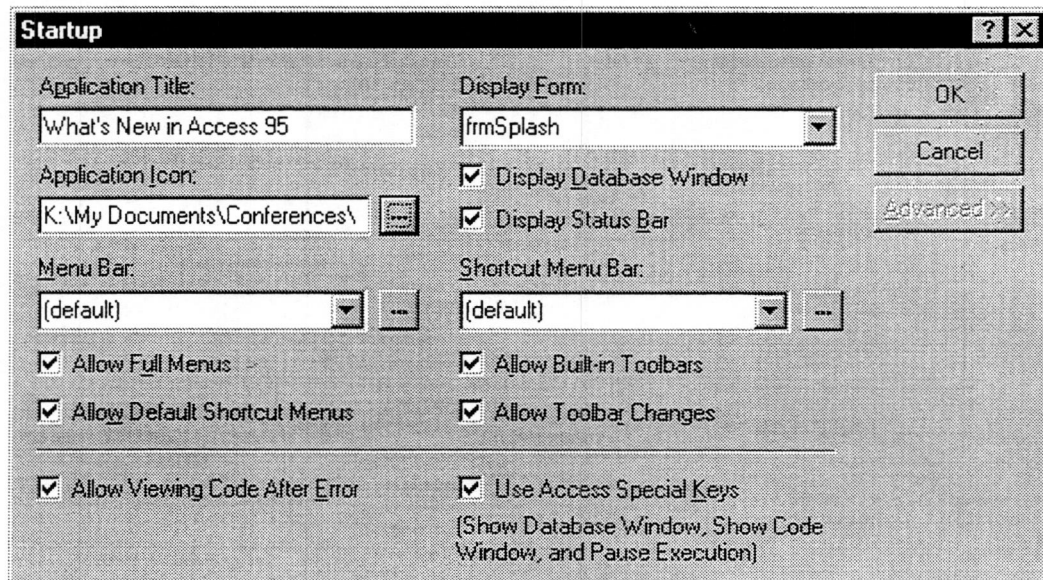


Figure 35. Startup properties for the sample database.

As you can see, Startup properties let you control not only the form and icon for the database, but also let you turn off some of the more dangerous options before the end users of your database exercise them.

TIP: There is no need to create an AutoExec macro in Access 95. Just use the Open Event of your Display Form to run whatever code you want to execute when the database is first loaded.

Database Splitter Wizard

If you have worked with Access on a network, you have seen the benefits of splitting a database into two parts. When you store all of your tables in one database and all other objects in another, you share data easily while not loading your network down with needless traffic to load form and report definitions. Access 95 includes a new Wizard to quickly convert a single database to two databases and then link the two databases back together.

To run the Wizard, select Tools\Add-Ins\Database Splitter.

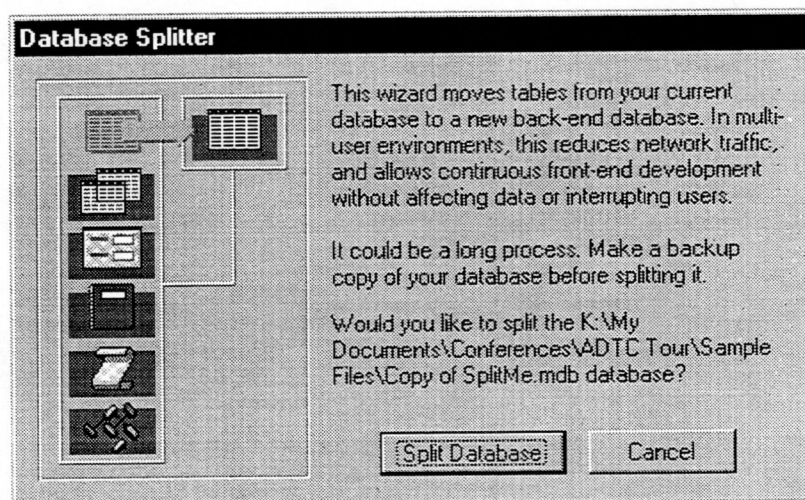


Figure 36. The Database Splitter Wizard.



Split a copy of the sample database to demonstrate how this Wizard works.

What's in the ADT?

If you are doing professional development with Access 95, you almost certainly want to purchase a copy of the Access Developer's Toolkit. This extra package includes:

- An unlimited redistribution license for the runtime version of Access 95.

- Printed copies of the Access Language Reference and DAO Reference.
- A Setup Wizard to create installation disks for your application.
- A selection of Custom Controls for Access forms.
- The Win32 API Viewer.
- Replication Manager and Transporter.
- Microsoft Help Workshop.

You'll learn about the replication tools and custom controls a bit later in the day. Right now, let's take a look at the Setup Wizard.

Improved Setup Wizard

When you purchase the ADT, you get a redistribution license for runtime versions of all of your Access 95 applications. The Setup Wizard is the tool that builds installation disk sets for these runtime versions. A runtime Access application is fully functional *except* that all of the design views of objects are permanently invisible. This means that your customers can run your application without buying Access, but they cannot create new applications.

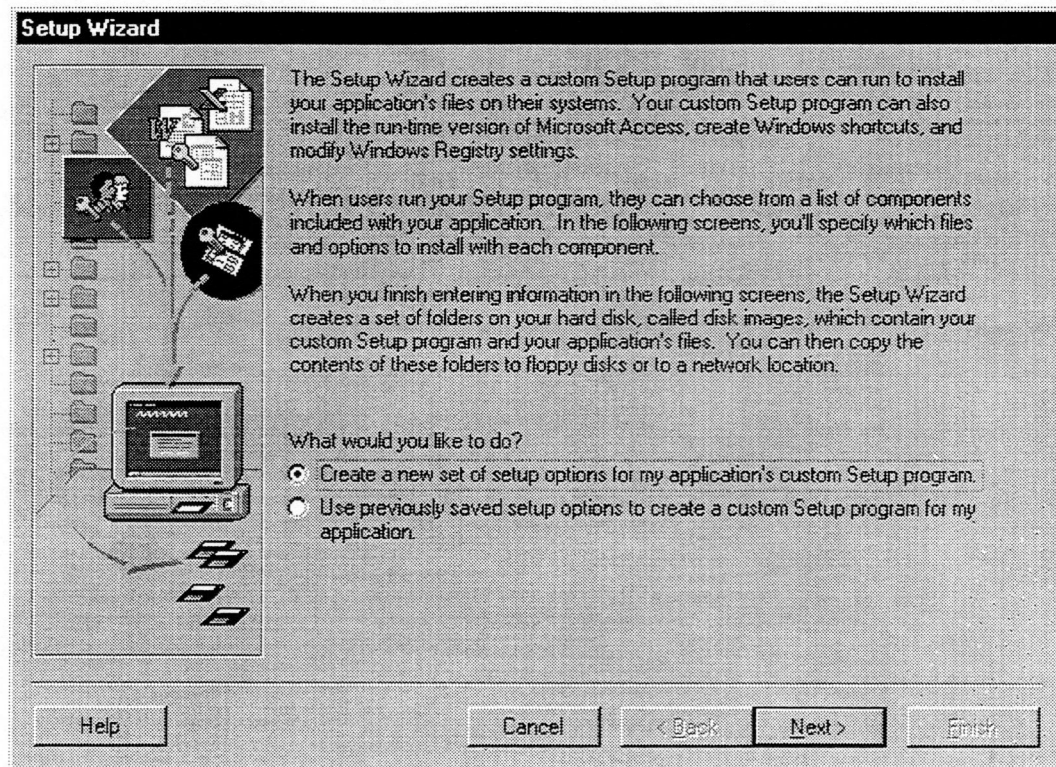


Figure 37. First screen of the Setup Wizard.

The Setup Wizard walks you through all the choices necessary to build setup disks. This includes:

1. Which files to include with your application.
2. What Windows 95 shortcuts to create when your application is installed.
3. What custom registry keys to install with your application.
4. What optional components to install with your application.
5. Which files to install in Typical, Compact, and Custom setups.
6. What to call your application.
7. What file to run when the setup is complete.
8. Whether to create diskettes or a network setup.



Use the Setup Wizard to make a redistributable copy of the sample application.

Form Design Improvements

Many of the new features in Access 95 center around creating new forms. There are so many new features, this course cannot possibly cover them all. This section focuses on the features that we find most interesting, but you are certain to find your own favorites, in addition to the items listed here. In any event, creating and working with forms and reports in Access 95 is much easier than in previous versions.

Creating Forms

Create a Bound Form Based on a Filtered Datasheet

You create a form or report based on a filtered datasheet (table, query, or other form), and that form inherits the filter of the original datasheet. When you save the form or report, Access saves the filter with the object, and it is still there the next time you open it.

TIP: Although Access does save the filter you set for the datasheet when you save the form or report, that filter is not active when you next open the object. To activate the filter, you need to either set the FilterOn property to True, or select the Tools|Apply Filter menu item or the Apply Filter toolbar icon.



Create a new form based on a filtered datasheet.

Tabbed Property Sheets

To be more consistent with the rest of the Windows 95 environment, Access 95 supports tabbed property sheets. Not only does this make Access *fit in* better with the Windows 95 environment, it makes it easier for you to find the properties you need. Figure 38 shows the new Property Sheet, with its new look.

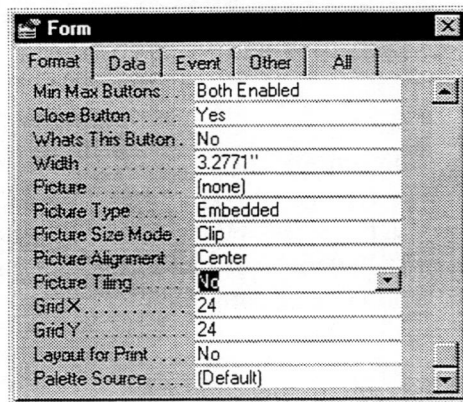


Figure 38. The new Property Sheet sports a new, tabbed look.

Cycled Properties

Programmers working in Visual Basic enjoy a feature that has eluded Access developers: the ability to double-click on properties in the Property Sheet. In Access 95, double-click on any property in the Property Sheet that is based on an enumerated list, and Access cycles through all the possible values for that property. For Yes/No fields, double-clicking switches between the two values. For multiple-valued properties, Access cycles through all the possibilities for the property.

Use AutoFormat to Set Up Your Forms

Access' AutoFormat feature allows you to apply a saved format (form and control properties, including the form's background bitmap) to any form in design view. Not only can you apply existing formats, but you can create your own named formats and apply these new formats to forms.

Use the AutoFormat feature to create a *look* for your company's applications: design a format, and apply it to all your forms and reports. Because you select a saved style when you create objects with the Wizards, use your saved formats for forms you create automatically, as well as those you create by hand.



Look at the Tabbed Property Sheet, Cycled Properties, and Autoformat: both using existing formats and creating new ones.

Default Controls for Fields

When you create a table, you have the option to specify a display control so that when you use the data, Access knows how you want it displayed. When you create a form or report based on this data, Access again uses the DisplayControl property of each field to create the correct control type.

This makes it possible for Access to create forms and reports that look just like you like them to: it not only uses the correct control type, but includes the correct lookups for list and combo boxes.

Control Morphing

If you have ever created a combo or list box, set all the properties accordingly, and then needed to change it to some other control type, you will appreciate this new capability. Access 95 allows you to convert an existing control into any different control type (within reason). It performs the type conversion, and maintains all appropriate properties.

TIP: You can also use control morphing programmatically: every control exposes its ControlType property (and Access provides constants representing all the possible values of the property). To change a control's type, just change the ControlType property for any control on an open form in design view.



Demonstrate default control types and control morphing—two features that save you tons of time.

Add a *What's This* Button

Access 95 allows you to place a *What's This* button on your forms' borders, right next to the Form Close button. If a user clicks on the *What's This* button, Access changes the mouse cursor into a question mark, and the user can click on any other control on your form. At that point, Access calls WinHelp and displays the help page you have associated with the control. Figure 39 shows a form with a *What's This* button.

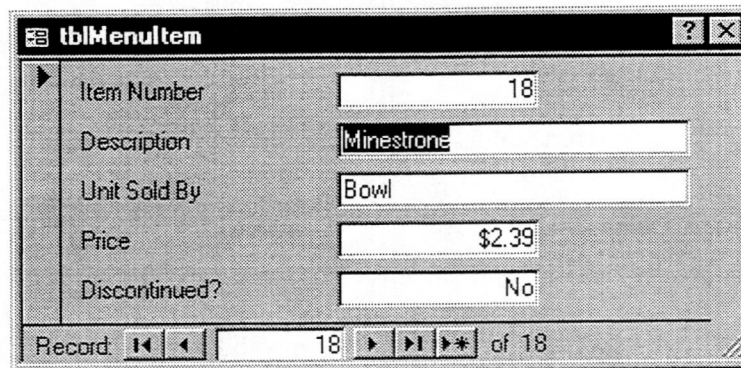


Figure 39. The What's This button allows you to provide context sensitive help for any item, selected or not.

To make use of the *What's This* button, you need to set the form's HelpFile property and the HelpContextID for each of the controls on the form. Access uses the form's HelpFile property to know which help file to load, and the HelpContextID property of each control tells Access exactly which page to load within that help file.

WARNING! You cannot have both the form's Minimize and Maximize buttons visible **and** have the *What's This* button visible. If you want to use the *What's This* button, you will not be able to maximize or minimize the form.



This demonstration shows how to add a *What's This* button to a form. To make things simple, just use a page from the Access help file to demonstrate the functionality.

New Control Properties

Once you create your form, you need to work with the control properties. Access provides some new properties you can use to customize your forms' controls (none of these properties applies to reports).

Control Tips

Access 2.0 provided tips that appeared as you hovered over a button with your mouse, but didn't make it possible for you to create the same sort of tips for your own controls. (Many people attempted to program this, with varying

results.) In Access 95, you no longer need to fight this battle: every control provides its ControlTips property, allowing you to add your own control tips for controls on your forms.

TIP: Do not go overboard with control tips. They are most appropriate for command buttons and perhaps other controls that require users to click on things. For normal data input controls, it is less obtrusive to use status bar text instead.

Shortcut Menus

Access 2.0 also provided right mouse-click, context-sensitive menus, but neglected to provide a way for developers to create their own menus. Again, developers cobbled together their own solutions to this problem. Access 95 makes this simple: hook a menu macro to a control's ShortcutMenu property, and a right-click on this control at runtime causes your menu to pop up.

Setting up the macros for shortcut menus requires the same effort as setting up any other menu within Access. Run the Menu Builder add-in to create the menus.

WARNING! Because of an omission in the runtime version of Access, you need to use the same sort of workaround for runtime applications' shortcut menus as you did in Access 2.0. That is, the built-in shortcut menu technology does not work in the current version of the Access 95 runtime version.

Allow Autocorrect

As part of its Office Compatibility compliance, Access provides AutoCorrect for controls on forms. There may be times, however, when you want to turn off AutoCorrect for one or more controls on your forms. Each control supplies an AllowAutoCorrect property that allows you to turn on or off the AutoCorrect capability of the control. You may decide to turn off AutoCorrect for proper name fields, so Access does not try to correct spellings of names for you.



Look at these three new properties: Control Tips, ShortCut Menus, and AutoCorrect.

Working with Controls

Using the New Image Control

In Access 2.0, if you needed to embed a static image on a form, your only choice was to use an unbound OLE object container. Unfortunately, this control carries with it a large amount of overhead that you just don't need if you're just displaying the image. The OLE object container allows you to edit the image as well as display it, and for static images, that's not an issue.



Access 95 provides a new control type—the Image control (its toolbar icon is displayed in the left column). This *lightweight* control uses far fewer resources than the OLE object control, which means it places a smaller strain on your computer, and allows your forms to load measurably faster. Any time you intend to display an image, but not allow it to be edited, use the Image control rather than the unbound OLE object.

Using the Format Painter



Just as in Word and Excel, Access provides a Format Painter (the toolbar button is shown to the left). The Format Painter copies formatting from the selected control, and allows you to apply that formatting to one or more other controls on your form. This functionality is only available via the toolbar, so do not turn off your toolbars if you want to use the Format Painter!

The table below includes a list of the properties that the Format Painter picks up and applies for you.

Properties Affected by the Format Painter	
BackColor	FontSize
BackStyle	FontUnderline
BorderColor	FontWeight
BorderStyle	ForeColor
BorderWidth	SpecialEffect
DisplayWhen	TextAlign
FontItalic	Visible
FontName	

Using the Format Painter is just one of the many ways to set the formatting for your controls, and it is also one of the simplest: taking only one action sets all fifteen of the listed properties for you.

In addition, just as in Word and Excel, double-clicking the Format Painter icon makes its actions persistent. Set the properties of as many controls as you like, until you press Esc to cancel the mode.

New Special Effects for Controls



To make it possible for your forms to include the Windows 95 *look-and-feel*, Access 95 provides new special effects for your controls. The floating palette shown at the left provides the six different options: flat, raised, sunken, etched, shadowed, and chiseled. If you have ever fought with the low-tech way Access 2.0 created its shadowed-look controls, you will be glad to finally have shadowing as a formatting option.

Allowing Nulls: Triple State Controls

Access has always had controls that allowed you to display and enter two-state data: check boxes, option buttons, and toggle buttons all allow you to select a Yes or No value. And they even allowed a third state—Null—if you had never entered data into the control. However, once you set the value, there was no going back.

Access 95 provides the TripleState property for each of these two-state controls. This makes it possible for you to set a control to be Yes, No, or Null. If you need to change the state of a Yes/No field to be “*I don’t know*,” you will appreciate this new property.

NOTE: Once you place these controls into an option group, Access removes the TripleState property. You can set the Option Group

itself to be Null (none of the internal options are selected), but none of the internal controls will supply the TripleState property.



This demo covers the Format Painter, the TripleState property, and some of the new control formats.

Working with Combo and List Boxes

Improved LimitToList Behavior

In Access 2.0, if you set the LimitToList property of a combo box to Yes, it was quite difficult to empty the combo box once you entered data. Because Null was not an allowable value, you could not set the combo box's value to be Null once it had been non-Null. The only two solutions involved asking the user to press Esc to clear out the data, or programming some mechanism to reset the value.

In Access 95, LimitToList is more sensible: if you set the combo box's value, and then want to set it back to being empty, just delete the value. The combo box accepts the value (Null), and you do not have to write a bunch of code in the NotInList event to work around this problem.

Multi-Select List Boxes

Access 2.0 did not allow you to select multiple items in a list box, even though users had long been accustomed to this behavior in Windows. Providing workarounds to this problem became a sort of "*cottage industry*," but nothing worked as well as a built-in multi-select list box.

Access 95's list box provides three multi-select options: none, simple, or extended. *None* corresponds to the original mode—you can only select a single item. *Simple* allows you to select multiple items by clicking on each item: another click on an item deselects it. *Extended* allows the list box to work the way Windows' list boxes have always worked: click to select an item; Ctrl+click to select multiple items; and click on one item, followed by Shift+click on a different item to select a contiguous range of items.

Once you select items, Access provides a number of methods you can use to find out how many, and exactly which, items are selected. ItemsSelected is a collection containing one item for each selected item in the list box.

Traversing this collection provides a list of the selected items. You can also use the Selected property of the list box: this property returns an array, one-to-

one with the items in the list box. Each element of the Selected property is either True or False, indicating whether an item is selected or not. If you want to know if a particular item is selected, use the Selected property. If you want a list of all the selected items, the ItemsSelected collection gives you the information you need.



Show the new **LimitToList** functionality, and **Multi-Select List Boxes**.

Working with Subforms/Datasheets

The New Subform Wizard

Creating subforms and subreports has never been easier! Access 95 provides a helpful wizard that nearly automates the process. Even if you have not created the necessary queries to fill the subform, Access does that for you. Figure 40 shows the output of a sample run with the SubformWizard.

OrderID	Quantity	ItemDescription	Price
1	1	Salad	\$1.99
1	1	Small Coke	\$0.50
1	5	Large Diet Pepsi	\$0.75
1	1	Large Diet Pepsi	\$0.75

Figure 40. The subform wizard can create form/subform combinations like this in just a few steps.



Create a subform from scratch, using the **Subform Wizard**.

Using the Subform Linking Builder

If you create subforms without the Wizard, and have a hard time linking subforms to their parent forms, the Subform Field Linker helps you out. (This only happens if you have not set up permanent relationships between tables in your application. Access cannot guess the relationships between the tables if you did not create fields with the same names in the underlying tables.) This builder provides lists of possible fields in the master and child data sets, and even suggests matches if you do not know which fields are to be used to connect the two forms.

Start the Subform Field Linker by clicking the Build (...) button next to the LinkChildFields or LinkMasterFields properties of the subform control. Figure 41 shows the Subform Linking Builder in action.

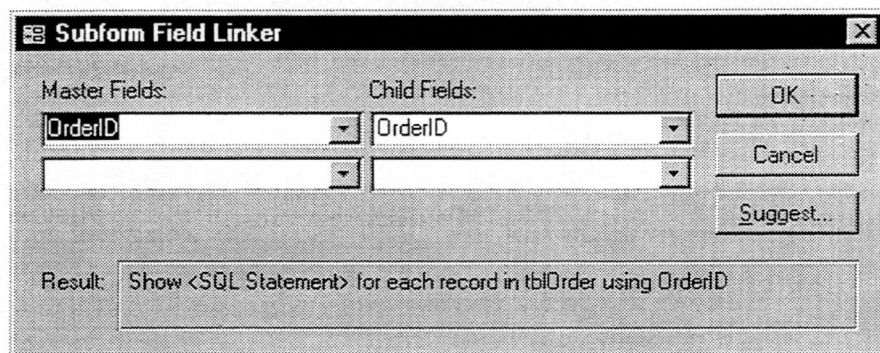


Figure 41. Use the SubForm Field Linker to link master and child forms.

Working with Selected Rows on a Datasheet

Access 95 now gives you the information you need to know which rows a user has selected in datasheet view. The new SelTop, SelLeft, SelWidth, and SelHeight properties allow you to always know which rows and columns of a datasheet are currently selected.

The sample form, frmOrders, calls a macro when you press Ctrl+S that displays the range of values currently selected. It uses the four properties to calculate the selected rows and columns currently selected in the subform. Figure 42 shows the sample form, with some rows and columns selected.

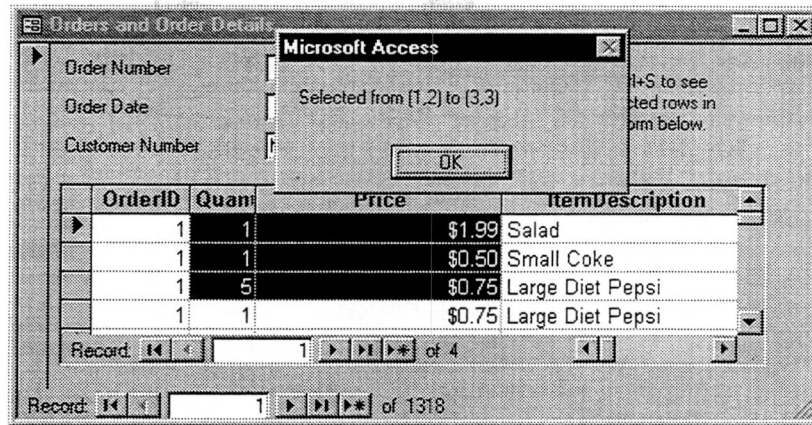


Figure 42. You can use SelTop property, etc., to find out what is selected on a datasheet.

This ability has been much requested in previous versions of Access. Developers often want to be able to tell which rows are selected in a datasheet, so their applications can react by operating on each of the selected rows.



Demonstrate how you can find out which rows and columns are currently selected, using the sample form frmOrders and its subform.

Advanced Form Properties/Events

Using the Filter and Sort Properties

In previous versions of Access, it was nearly impossible to find out how a form had been filtered by the user: if you wanted to open a report showing the filtered rows on a form, you would be in for a lot of trouble. Now, Access 95 makes this trivial: it supplies the FilterBy and FilterOn properties of forms that you use to find out how a form's rows are filtered, and whether the filter is currently active.

In addition, use the form's OrderBy and OrderByOn properties to find out if the user added any additional sorts to the form's recordset, and whether they are active.

TIP: If you open a saved form that has its Filter property set, you need to set the FilterOn property to True to make it active. However, the same does not apply to the OrderBy property: it is active when you first open the form.



To see the **Filter** and **OrderBy** properties, look at a form created from a filtered recordset.

The Cycle Property

Access 2.0 provided no simple method for keeping users on the current row when editing a form. Once they moved past the last control on a form, or moved backwards before the first control, Access automatically moved them to the next or previous row in the data. Many developers wanted some way to *force* users to stay on the current row, wrapping the focus back around.

Access 95 forms provide a **Cycle** property, which allows you to cycle the focus around All Records (the same as Access 2.0 behavior), Current Record, or Current Page. Cycling through the Current Record keeps the focus on a single row of data, and Current Page keeps the focus on a single page of a multi-paged form. The two new options allow you much greater control over the user's navigation on forms.

Using the KeyPreview and Key Event Properties

Many developers want complete control over the way forms react to keystrokes, and Access 95 makes this possible. In Access 2.0, a form that contained controls (and most do, of course) could not receive any keystrokes. If you wanted to react to a keystroke on a formwide basis, you had to either use **AutoKeys**, or write code attached to every control's **Key** events.

Access 95 forms provide a new **KeyPreview** property. Setting this property to *True* tells Access to send all keystrokes to the *form* before any controls react to them. In this way, your form reacts to keystrokes itself.

For example, one common problem developers face is trying to disable the **PageUp** and **PageDown** keys—effectively keeping users from moving from one record to another except under program control. In Access 2.0, you were required to open the form in dialog mode if you wanted this level of control, but that limited the usefulness of the form. Now, use the **KeyPreview** property of a form to trap the **PageUp** and **PageDown** keys and disregard them, if that is what you need to do.



Demonstrate how you can use the **Cycle and **KeyPreview** properties to take control over how and where your users move on your forms.**

OLE Support

As with all the other Microsoft products, support for OLE (Object Linking and Embedding) becomes more and more prevalent with each release of the product. The following sections outline some of the new OLE-based features in Access 95.

Bundled OLE Controls

OLE controls allow developers to extend their Access 95 forms and reports with pre-built OLE components. Access 95 supports both 16 and 32-bit OLE controls. Depending on where you look, you will find many new OLE custom controls that Access 95 can use. The benefits of using OLE controls are:

- They are tested, debugged, and ready for use.
- They can be tightly integrated with your application, residing on the same forms as other controls, and can bind directly to your data.
- They can be shared between development environments (i.e., VB4 and Access 95).

In the Access Package

The Access 95 package includes a data-aware Calendar control. This control has all the attributes of a calendar: it displays the month, day, and year for any date. But this calendar is special—it also binds to an Access data field. This means the calendar displays the value of a date field, and when the user makes a change to the calendar, Access also updates the data in the underlying table. Binding the calendar control to your data requires no code: just set the properties correctly, and you are all set.

In addition, Custom Control properties are now available from the standard property sheet. Previously, you had to go to two places to edit the properties specific to a custom control. Now, all the properties that are specific to the custom control are also available from the right-click menu on the control.



Take a look at the data-aware Calendar custom control.

In the ADT

The Access Developer's Toolkit provides a number of other OLE controls. In addition to a number of common Windows 95 controls, the ADT includes other useful interface elements. The table below lists all of the controls included in the ADT.

Custom Controls in the ADT	
CommonDialog	Slider
Data Outline	SpinButton
ImageList	StatusBar
ListView	TabStrip
ProgressBar	ToolBar
RichTextBox	TreeView

These controls are a subset of the controls provided in Visual Basic 4. They are invaluable in creating applications that fit the Windows 95 *look and feel*. Figure 43 shows the sample form, combining three of the OLE controls.

NOTE: Although many of the controls available for VB4 programmers work in Access 95, many do not. Two types of controls do not work in Access 95: those that are bound to complex data sources (that is, are bound to a row in a table, as opposed to a column), and those that are meant to contain other controls. If in doubt about a retail custom control, ask the vendor if it works in Access 95.

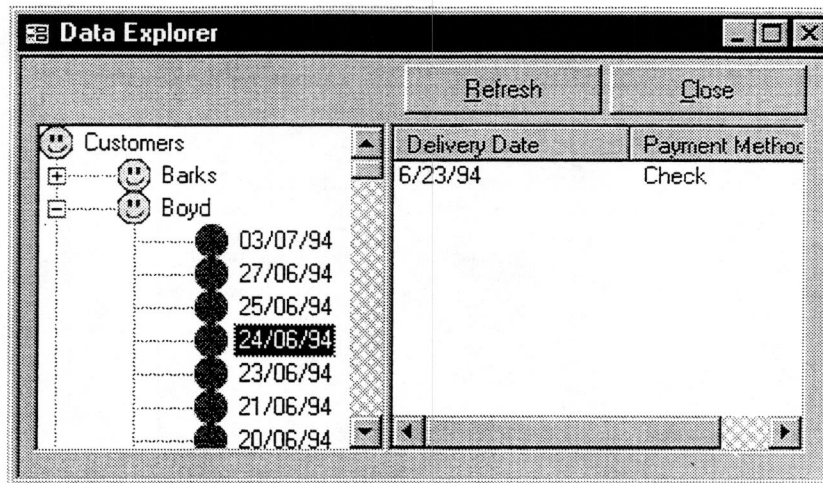


Figure 43. TreeView, ListView, and ImageList controls combined together.



Take a look at three of the Windows 95 common controls: the ListView, ImageList, and TreeView controls. ImageList provides the images, TreeView provides the left-hand pane, and ListView provides the right-hand pane of this data viewer.

Object Browser/References Dialog Box

VBA is nicely suited to OLE programming. The References dialog box and the Object Browser allow the VBA programmer to cross the boundaries between applications.

The References dialog box, shown in Figure 44, lets you *declare* the other object libraries and databases that you need to reference in your application. If you work programmatically with Excel, PowerPoint, or whatever, you need to declare a reference to that application's type library, so you can use its constants, functions, and types in your VBA code.

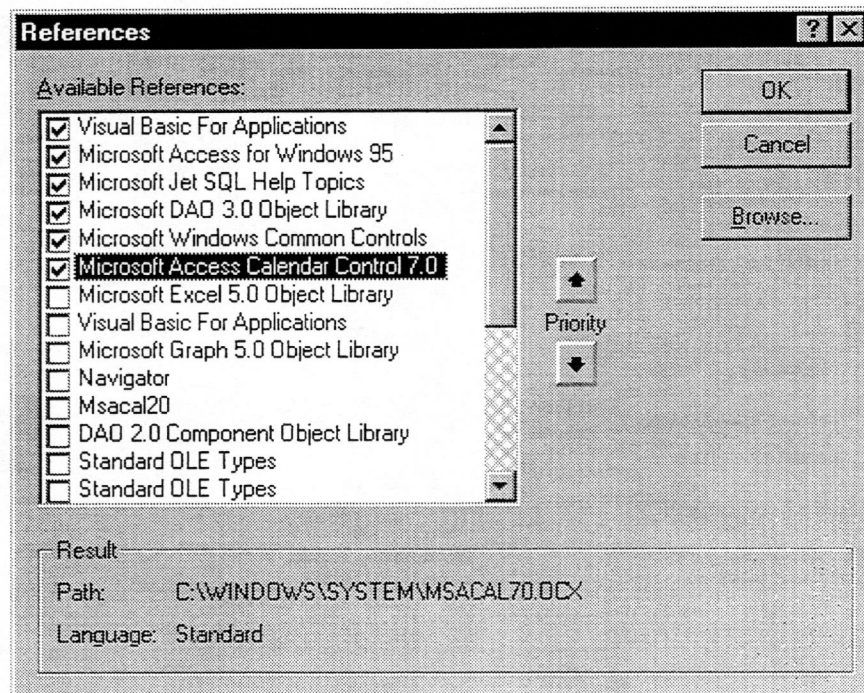


Figure 44. The References dialog box allows you to “connect” with other applications.

Once you set up a reference to the external library, use the built-in Object Browser to view and learn about the OLE objects. You can find objects, constants, data types, methods, and properties, and the Object Browser allows you to paste code directly into your application.

You can use the Object Browser to investigate object models for any OLE server for which you set up a reference. Use it to help build code to call any OLE server—Jet, VBA, or any other Office application.

Figure 45 shows the Object Browser in action. Once you select an object, method, or property, you can take any of the following actions:

- Look up the object in online help.
- Dig deeper into the hierarchy.
- Paste the syntax for the object into your code.
- Select another object to browse.

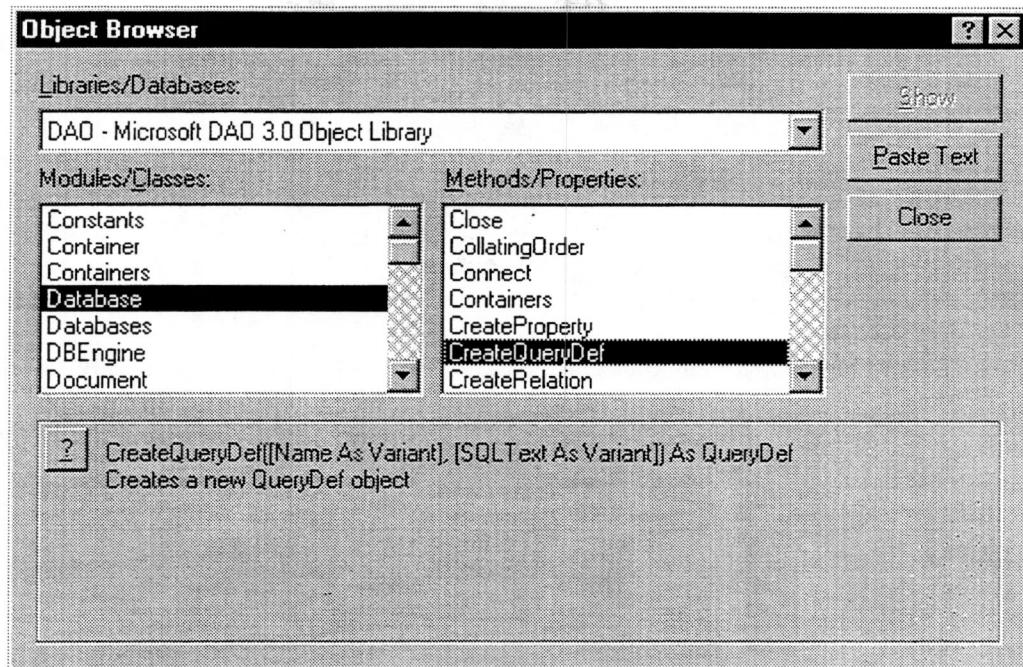


Figure 45. The Object Browser helps you dig through any OLE server's objects, methods, etc.



Add a new reference to the current database, and use the Object Browser to investigate the OLE server, using it to paste code into the sample application.

OLE Automation Server

Although custom controls are enhanced for Access 95, the ability to control Access using OLE Automation is totally new. Use Access as an OLE Automation object to run Access forms, queries, and reports from other applications: Excel, Project, and VB4, among others. Access 95 also continues to be an excellent OLE Automation controller. It manages any application that functions as an OLE Automation server, including Excel, Word, Graph, PowerPoint, Schedule+, and even several non-Microsoft products.

Anything you can do in Access programmatically, you can do from any other application using OLE Automation. The DoCmd object gives access to all the macro actions, and DAO and the Application object provide access to all the objects. This makes it possible for other applications to run reports, open

forms, retrieve data, perform action queries: use your imagination for more uses of OLE Automation with Access.

Your applications need to be aware whether they are being run from the Access user interface, or from an OLE Automation controller. The Microsoft Access 95 Application object provides a property (`Application.UserControl`) that tells you whether the user is in control. Access also supplies the `Visible` property of the Application object (read-only unless `Application.UserControl` is `False`) that allows you to make the application window visible or invisible.

One common use for OLE Automation is to run reports from other applications. The Microsoft Access report writer is the best in the business, and it is far easier to have Access run a report than to attempt to duplicate the functionality in Excel or Word.



This demonstration shows a simple example of using OLE Automation to cause Access to display a report, from Excel.

What's New for Coders?

Reproduced from Microsoft® Access® 95 Developer's Handbook™ by permission of Sybex Inc., ISBN 0-7821-1765-1 Copyright © 1996, SYBEX, Inc. All rights reserved. For further information please contact info@sybex.com or 1-800-227-2346."

VBA? What's That?

Visual Basic for Applications (VBA) is one of the largest additions to Access 95. VBA was originally introduced into Excel 5.0 as a cross-product application language. In this revision of the Office products, Access, Excel, and Project share the same language engine with Visual Basic, the development environment. This course refers to the shared language engine as Visual Basic: where you see the term *Visual Basic* in this document, it is in reference to the language, not the product. (Of the Microsoft products that include a Basic-like programming language, only Microsoft Word includes its own variant of Basic (WordBasic), and even that will change in an upcoming version.)

This session presents many of the features of Visual Basic that differ from corresponding features in Access Basic (sometimes referred to as Embedded Basic). From the programming environment, to the syntax of the language, to the details of compilation, Visual Basic has many changes that you need to be aware of as an Access developer. You can find all the examples used in this section in the database NEWSTUFF.MDB. Take the time to dig into the new features of Visual Basic: start with the examples presented here, and build from there. Or start from scratch, using the techniques you find here.

The VBA IDE

The Access IDE provides the platform you use to create Visual Basic code. Because you cannot reasonably replace it with any other editor, its features and abilities play a major part in how you develop applications. This section discusses the various new features in the development environment.

Color-Coded Editing

The first time you start writing Visual Basic code, you will notice some major changes in the programming environment. Color coded text, one of the most-requested omissions from Access 2.0, makes it clear just by glancing at your code which portions are comments, which are keywords, and which are errors. After working with Visual Basic for a while, programming in Access 2.0 seems very monochromatic! Using the Tools|Options|Module dialog box, you can control the foreground and background colors of code window, selection, syntax error, breakpoint, next statement, comment, keyword, and identifier text.

Like most programmers, you probably worked with syntax checking turned on in Access 2.0, because with that feature enabled, Access tells you when you type incorrect syntax. For example, if you entered half a line of code, then needed to move to some other place to find an object name, Access assumed that you had made a syntax error, and gave you an alert. In Visual Basic, you can turn off syntax checking (see the Tools|Options|Module|Auto Syntax Check option) and let color-coded editing show your errors. This way, if you leave a line of code half-edited, Access does not interrupt you with an alert, but just displays the unfinished line in the error text color. Once you try this option, you will never want to go back to the old way!

Customizable Font

You can adjust the font and size of the text used in the module editor. This is useful both for normal editing (where you might want to use a small font to see wide lines of code), and for presentations, where your audience will at last be able to see the module editing font. If you work regularly with a high-resolution screen driver, you may find that using a large font saves your eyesight.

Line Continuation Character

Lines of code in Basic can become quite long, and can extend past the edge of your editing window. Visual Basic provides a line-continuation character (it is actually four characters: space, underscore, carriage return, and line feed) that allows you to divide text anywhere you like, except in the middle of a quoted string or keyword. Code that used to look like this:

```
strSQL = "SELECT DISTINCTROW tblCustomer.LastName, "  
strSQL = strSQL & "tblCustomer.FirstName, "  
strSQL = strSQL & "tblOrder.OrderDate, "  
strSQL = strSQL & "tblOrderDetail.Quantity "  
strSQL = strSQL & "FROM tblMenuItem INNER JOIN "  
strSQL = strSQL & "((tblCustomer INNER JOIN "  
strSQL = strSQL & "tblOrder ON tblCustomer.CustomerID "  
strSQL = strSQL & "= tblOrder.CustomerID) "  
strSQL = strSQL & "INNER JOIN tblOrderDetail "  
strSQL = strSQL & "ON tblOrder.OrderID = "  
strSQL = strSQL & "tblOrderDetail.OrderID) "  
strSQL = strSQL & "ON tblMenuItem.MenuID = "  
strSQL = strSQL & "tblOrderDetail.MenuID;"
```

can now be written like this:

```
strSQL = "SELECT DISTINCTROW tblCustomer.LastName, " & _  
"tblCustomer.FirstName, tblOrder.OrderDate, " & _  
"tblOrderDetail.Quantity FROM tblMenuItem " & _  
"INNER JOIN ((tblCustomer INNER JOIN " & _  
"tblOrder ON tblCustomer.CustomerID = " & _  
"tblOrder.CustomerID) INNER JOIN tblOrderDetail " & _  
"ON tblOrder.OrderID = tblOrderDetail.OrderID) " & _  
"ON tblMenuItem.MenuID = tblOrderDetail.MenuID;"
```

Full-Module View

Access 2.0 limited your module editing to a single procedure (or two procedures, if you used a split window). If you wanted to cut and paste large chunks of your code, the simplest solution was to save the code as text, and work with that text file. Access 95 allows you to either view your code the old

way or to view your modules as a continuous flow (Tools|Options|Module|Full Module View). You can even decide whether or not to view the procedure separator bar (Tools|Options|Module|Procedure Separator). Figure 46 shows these two options on the Tools|Options dialog box. If you display your modules in Full Module View, it is easy to select code in more than one procedure, or to view multiple procedures.

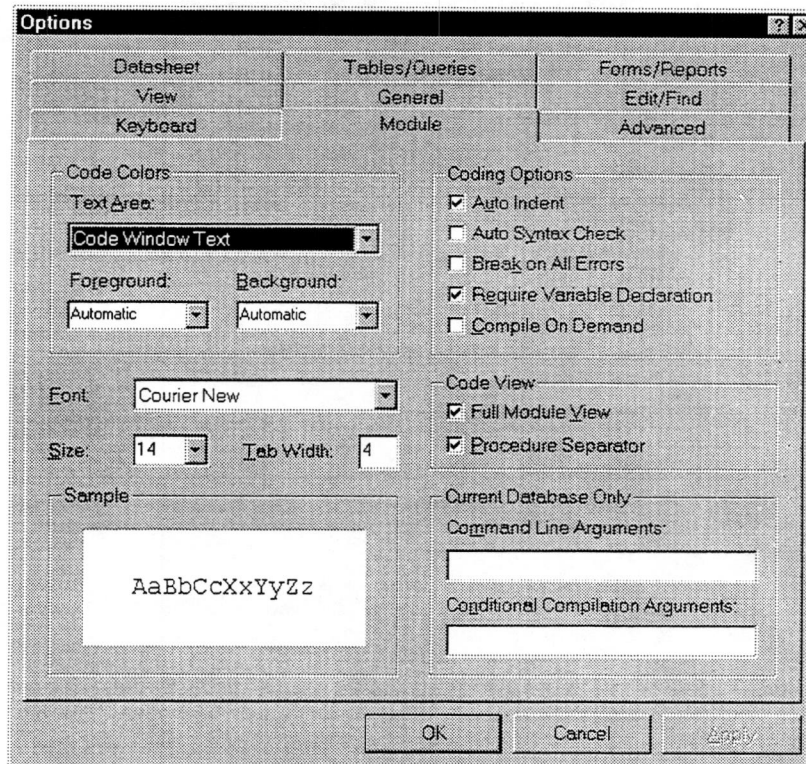


Figure 46. The Tools|Options|Module dialog box allows you to set Full Module View and other module options.



Try out some of the new IDE features, including color-coded editing, line-continuation character, and Full Module View.

Watch Points and Conditional Breakpoints

To help you debug your applications, Access 95 provides for watch points and conditional breakpoints. A watch point allows you to continually watch the value of a variable or expression in the Debug Window. This way, rather than

using Debug.Print or querying the value of an expression in break mode, a watch point constantly updates the value of the expression for you. You can also select a variable or expression, and use the Tools\Instant Watch menu item to add that expression to the Watch window (the upper pane of the Debug Window). Figure 47 shows the Debug Window with a watch point set.

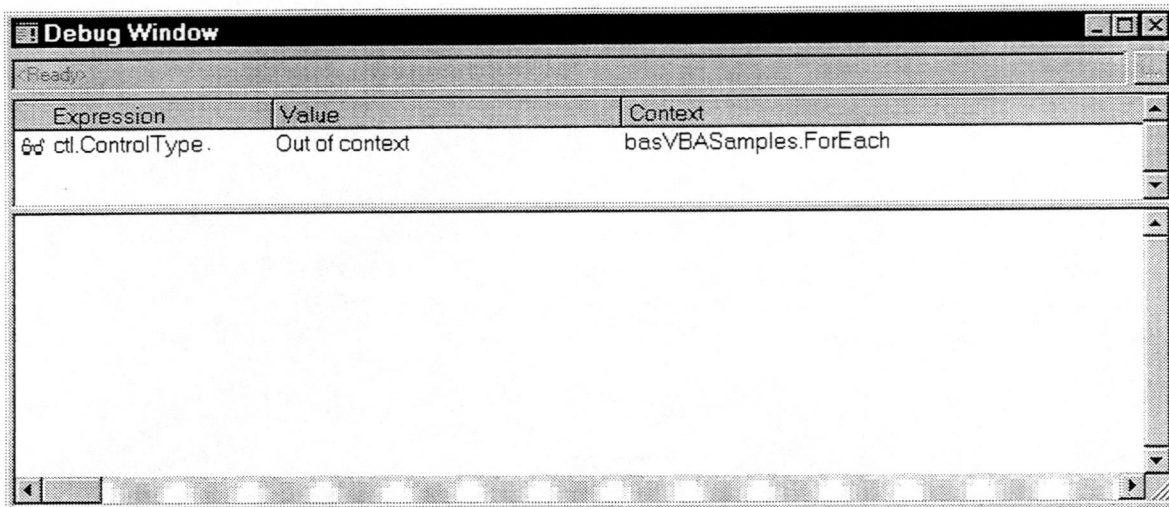


Figure 47. Watch points allow you to continually watch the value of variables.

Conditional breakpoints work like normal breakpoints except the code only stops at the line if a condition has been met. As with watch points, you specify an expression, and tell the code to stop if the expression changes, or when the expression becomes true.



Take a look at how Conditional Breakpoints and Watchpoints can aid in your debugging.

Environment Changes

The change from Access Basic to Visual Basic not only affected the language itself, but many of the surrounding parts of Access. This section discusses differences that are not strictly language-related, but are more a part of the environment (Access) itself.

The DoCmd Object

Access 95 is an OLE Automation server, and, as such, must expose its object model for other applications to use. Many activities in Access 2.0 require macro actions that in turn require the DoCmd keyword. The only way to execute these actions from outside Access is as methods of an object. Therefore, in Access 95, DoCmd has become an object, and all the macro actions are methods (subroutines) of that object.

```
DoCmd OpenForm "frmSampleForm"
```

In Access 95, the sample command is:

```
DoCmd.OpenForm "frmSampleForm"
```

Access 95 performs these changes for you when you convert your database from Access 2.0 format.

Conditional Compilation

Visual Basic allows you to insert conditional compilation commands, telling Access whether or not to compile certain areas of your code. This provides you with a mechanism for including debugging code that you “comment out” when you distribute your applications, or for using the same code on different platforms.

The allowed directives are #If, #Else, #ElseIf, #End If, and #Const. The #Const directive allows you to create module-private constants controlling the compilation of your code:


```
#If DEBUG Then
    ' Use this code for debugging only
#Else
    ' Use this code if not debugging
#End If
```

You normally set the value of conditional compilation constants in the Tools|Options|Modules dialog box. Constants created there are available to #If directives anywhere in your application. You can also set the constant locally, using #Const. Figure 48 shows the Tools|Options dialog box, with the constant set.

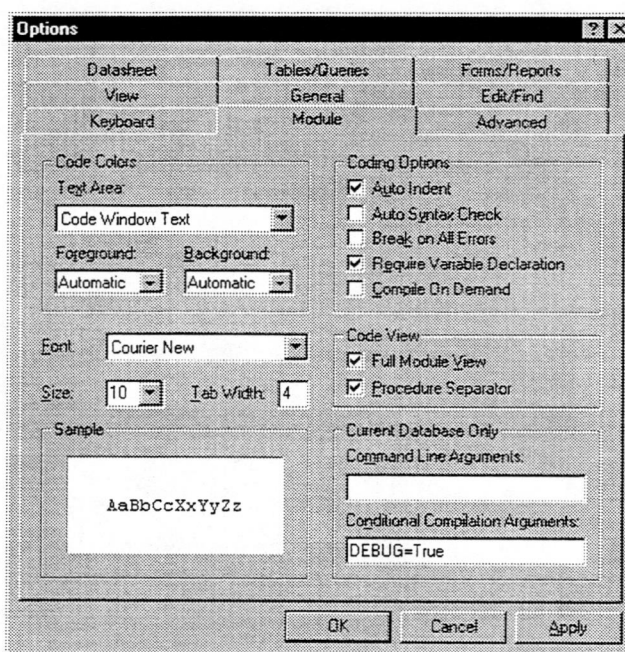


Figure 48. Use the Tools|Options dialog box to set conditional compilation constants.

Search All Modules

If you ever needed to search through all modules (including form and report modules) for a procedure or keyword, you know how difficult that could be in Access 2.0 (you really had to load each and every form and report individually to get their modules loaded as well). Visual Basic provides the functionality to search through all the modules in the current database, whether or not they are currently loaded.

Figure 49 shows the new Find dialog box, with the *Current Database* option selected.

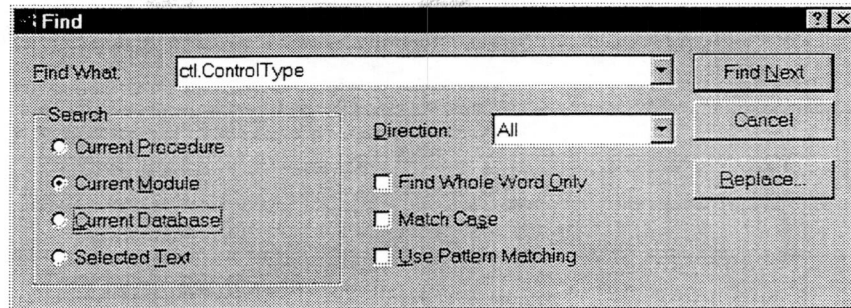


Figure 49. The new Find dialog box allows you to search through all the modules in the current database.

Language Changes

Some of the most interesting parts of Visual Basic are the new constructs and options it adds to Access Basic. The following sections outline some of the new language features, with examples of each.

Built-in Constants

Visual Basic provides a host of constants that make your coding life simpler. You do not have to import or type MsgBox constants anymore, and the Miscellaneous Constants table certainly makes your coding simpler—no more declaring a constant for Chr\$(13) and Chr\$(10)! The table below lists some of the new miscellaneous constants.

Constant	Value	Meaning
vbBack	Chr\$(8)	Backspace
vbCr	Chr\$(13)	Carriage return
vbCrLf	Chr\$(13) and Chr\$(10)	Carriage return and line feed combination
vbFormFeed	Chr\$(12)	Form feed
vbLf	Chr\$(10)	Linefeed
vbNullChar	Chr\$(0)	Null character
vbNullString		String whose value is 0 for API calls
vbTab	Chr\$(9)	Tab
vbVerticalTab	Chr\$(11)	Vertical Tab

For Each...Next

Access 2.0 allowed you to loop through all the members of a collection, but only if you knew beforehand how many elements the collection contained. You needed code like this (from the frmSampleForm's module) to list all the controls on a form, using the old technique:

```
' The old way...
Dim intI As Integer
Dim ctl As Control

For intI = 0 To Me.Controls.Count - 1
    Set ctl = Me.Controls(intI)
    Debug.Print ctl.Name, ctl.ControlType
Next intI
```

Using the new technique, you can simplify this a bit:

```
' The new way...
Dim ctl As Control
For Each ctl In Me.Controls
    Debug.Print ctl.Name, ctl.ControlType
Next ctl
```

You no longer need to know the maximum number of elements in the collection, and you no longer have to point a variable at the object in question (using the Set statement)—the loop does that for you.



This demonstration shows you how to use For Each...Next.

Some fine points to consider:

- The “loop” variable can be an object that matches the definition of the collection (as in the previous example) or it can be a Variant variable.
- Use For Each...Next with any collection (built-in or user-defined) as long as the collection does not contain user-defined types or fixed-length strings.

- Use For Each...Next to retrieve data from arrays as if they were collections. The same data type restrictions apply as in the previous bullet. However, you must use a Variant-type variable to loop through the array no matter what the data type of the elements of the array. This frees you from worrying about calling LBound and UBound to walk arrays of unknown size, just use For Each...Next to loop through all the elements.

With...End With

Visual Basic allows you to work with multiple properties, methods, or objects of a specific object using the With...End With syntax. If you used code like this in Access 2.0:

```
Forms!frmSampleForm.Width = 1400
Forms!frmSampleForm.Caption = "This is a test"
Forms!frmSampleForm!cmdEnumerate.Caption = _
    "Enumerate Controls"
```

you could now replace it with code like this:

```
With Forms!frmSampleForm
    .Width = 1400
    .Caption = "This is a test"
    !cmdEnumerate.Caption = "Enumerate Controls"
End With
```

Access treats any expression that begins with a "." or a "!" within the With...End With block as part of the object referred to throughout the entire block.



For this demonstration, we'll look at using For Each...Next along with With...End With to loop through controls on a form. The example is the routine ForEach in basVBASamples.

Named Parameters

Access Basic required you to use positional arguments when calling procedures. That is, if the procedure expected 7 parameters, and you needed to set the first, sixth, and seventh, you needed to insert commas as placeholders between the parameters. For example, to open a form in dialog mode and pass to it an OpenArgs parameter, you needed code like this:

```
' See NamedParameters in basVBASamples
' Call OpenForm the old way. Count those commas!
DoCmd.OpenForm "frmSampleForm", , , , , _
    acDialog, "Smith"
```

Visual Basic supports named parameters, allowing you to specify the parameter names and values, not worrying about the exact position of the arguments. The previous example, using named parameters, looks like this:

```
' Call OpenForm the new way. No more counting commas!
DoCmd.OpenForm FormName:="frmSampleForm", _
    WindowMode:=acDialog, OpenArgs:="Smith"
```

In addition, use this technique when calling your own procedures, making their calls self-documenting. For example, if you have a procedure named `HandleItems`:

```
Sub HandleItems(frm As Form, ForeColor As Long, _
    BackColor As Long, ShowIt As Boolean)
    Dim ctl As Control

    For Each ctl In frm.Controls
        With ctl
            If .ControlType = acTextBox Then
                .ForeColor = ForeColor
                .BackColor = BackColor
            End If
        End With
    Next ctl
    frm.Visible = ShowIt
End Sub
```

Call it like this (note that the order of the parameters in the calling list does not matter if you use named parameters; you do not have to use named parameters for all the items, just the last portion of the list):

```
HandleItems Forms!frmSampleForm, ShowIt:=True, _  
    ForeColor:=255, BackColor:=0
```

Optional Parameters

Visual Basic allows you to write procedures that accept optional parameters. The optional parameters must appear at the end of the procedures declaration (that is, once the Optional keyword appears in the declaration, all further parameters must also be optional). All optional parameters must be Variants, and you cannot have any optional parameters if you use the ParamArray option. Use the IsMissing() function to determine whether or not a parameter is missing, and react accordingly in your code.

Optional parameters allow your functions to work like built-in Visual Basic functions. If the caller supplies a value, use it. If not, use a default value instead. For example, the following function emulates the MsgBox function, requiring you to specify the text for the message box, but allowing you to omit the buttons (it uses "0" if you do not specify a value) and the title (it uses "MsgBox Test" if you do not supply a title). You'll find this function in basDeclare.

```
Function MsgBox2(Prompt As String, _  
    Optional Buttons As Variant, _  
    Optional Title As Variant)  
    If IsMissing(Buttons) Then  
        Buttons = 0  
    End If  
    If IsMissing(Title) Then  
        Title = "MsgBox Test"  
    End If  
    MsgBox2 = MsgBox(Prompt, Buttons, Title)  
End Function
```

Any of the following function calls work with MsgBox2:

```
varRetVal = MsgBox2("Hello!", 512, "Title Here")  
varRetVal = MsgBox2("A Simple Example")
```

```
varRetVal = MsgBox2("A Third Example", , _  
    "This is the title")
```



To try MsgBox2 (in basDeclare), call the function with different parameters (but remember that the first one is required).

Important topics to note:

- The first parameter is not optional, and Access requires you to pass at least a string value for the message box.
- Once you specify an optional parameter, the rest of the parameters must also be optional.
- All optional parameters must be Variants.

ParamArray

Using the ParamArray keyword in a procedure declaration allows you to pass an optional array of Variants. Its size can be arbitrary, so you can call the function with a different number of arguments each time.

For example, if you wanted to write a function to find the minimum value of a list of numbers, you could use ParamArray to allow you to pass any number of values to the function (see basDeclare for this example):

```
Function MinValue(ParamArray avarValues() As Variant) _  
    As Variant  
    Dim varValue As Variant  
    Dim varMinValue As Variant  
  
    ' Assume that none of the input values are numeric.  
    varMinValue = Null  
  
    ' Loop through each element of the array.  
    For Each varValue In avarValues  
        ' Only do this if the element is a number  
        ' of some sort.
```

```
    If IsNumeric(varValue) Then
        ' The first time around, varMinValue is
        ' Null. On the first pass, just assign the
        ' first element of the array to the min
        ' value. On all other passes, compare and
        ' assign as appropriate.
        If IsNull(varMinValue) Then
            varMinValue = varValue
        Else
            If varValue < varMinValue Then
                varMinValue = varValue
            End If
        End If
    End If
Next varValue
MinValue = varMinValue
End Function
```

To call this function, just pass it a comma-delimited list of values:

```
varMinValue = MinValue(-300, "Hello", Null, -454, 1)
' or
varMinValue = MinValue(1, 2, 3)
```

Some points to consider:

- ParamArray must be an array of variants.
- ParamArray argument must be the final argument.
- It cannot be optional.



To try this demo, open `basDeclare` and call `MinValue`, passing it various sets of parameters, as shown in the previous examples.

User-Defined Collections

Visual Basic allows you to create your own collections of variables and objects (you cannot create a collection of user-defined data types, but that is

your only restriction). Methods of collections allow you to add, delete, and retrieve items. Collections can even contain other collections. They are especially useful when you need to maintain a list of items whose size you cannot know until runtime: rather than continually redimensioning an array to hold items as you come across them, use the Add method of a collection to add the items as you need more space. You can also specify a unique key value for each item instead of the position in the collection for retrieving the item.

The following example requests names from you, adding each to a collection, until you enter a blank name, at which point it displays the entire list of names in the collection. Note that because this example uses a collection instead of an array, it can easily add items one at a time without redimensioning:

```
Sub TestCollections()  
    ' A very simple user-defined collection example.  
  
    Dim colNames As New Collection  
    Dim strName As String  
    Dim strOut As String  
    Dim varName As Variant  
  
    ' You could also have said:  
    ' Dim colNames As Collection  
    ' Set colNames = New Collection  
  
    Do  
        strName = InputBox("Enter a name _  
            (or leave blank to stop):")  
        If Len(strName) > 0 Then  
            colNames.Add Item:=strName  
        End If  
    Loop Until Len(strName) = 0  
  
    ' Now display the list  
    strOut = "Names in the collection: " & vbCrLf  
    For Each varName In colNames  
        strOut = strOut & vbCrLf & varName  
    Next varName  
    MsgBox strOut  
End Sub
```




For this demonstration, we'll try running `TestCollections` (in `basCollections`). This example requests names until you enter a blank one (press `Return`).

You'll find another example of user-defined collections in the section dealing with multiple instances of forms, later in the session.

New Data Types

Visual Basic provides new data types that make your code more robust and easier to read. Specifically (but not exhaustively):

- *Date* data type can contain a date/time value.
- *Byte* data type can contain a single byte.
- *Boolean* data type can contain a Yes/No value (-1/0).
- The *Object* data type can now refer to any object. This way, you can write a procedure that accepts an `Object` parameter, and send to it a reference to a form, report, control, or whatever you want. This makes it a lot easier to write generic functions that work with any kind of object.
- *Variants can now contain arrays*. That is, distinct from an array of variants, you can assign an entire array into a variant. The `IsArray()` function can check whether the variant contains an array or not.

Scoping Changes

In Access 2.0, variables and procedures were either global or private. That is, any object was either available only within its module, or globally throughout Access. In form and report modules, you could not create global objects.

Visual Basic offers two scoping levels: `Public` and `Private`. Using the `Public` keyword in form or report modules allows you to create procedures and variables that can be referenced from outside that module. All event procedures are `Private`, by default, but you can change them to `Public` and call them from wherever you like. In global modules, all variables and procedures are `Public` by default, but you can make them private using the `Private` keyword.

For example, the `cmdEnumerate_Click` event procedure in `frmSampleForm` has been changed to be a `Public` procedure. Once that form is open in form mode, you can call `cmdEnumerate_Click` from the Debug Window by typing:

```
Form_frmSampleForm.cmdEnumerate_Click
```

Because the name of the module is `Form_frmSampleForm`, you reference the procedure by typing the name of the module, a dot, and then the name of the procedure to call. You can also use the syntax:

```
Forms!frmSampleForm.cmdEnumerate_Click
```

Either way, you can run this Public procedure from another form, from any other procedure, or from a query or macro. In Access 2.0, programmers battled with ways to execute procedures in form and report modules: now it's simple!

The sample form, `frmSampleForm`, also includes a public variable, `PublicVar`, whose value you can retrieve or set, using the syntax:

```
Debug.Print Forms!frmSampleForm.PublicVar
```

or

```
Forms!frmSampleForm.PublicVar = 12
```

Changes to MsgBox

You can finally add your own help to message boxes! The `MsgBox` function supports two new optional parameters, `Helpfile` and `Context`:

```
MsgBox(prompt[, buttons][, title][, helpfile, context])
```

that allow you to specify a help file and a context ID for the page you want to display if the user sees this message box. If you specify one of those parameters, you must specify the other.

In addition, `MsgBox` supports a syntax that allows you to format part of the message in bold, possibly converting the prompt into a specialized "*Solution*" format. This technique requires two "@" signs with two or three possible pieces of text. The example procedure `ShowMsgBox` from `basVBASamples` shows the three possible outcomes of using "@" in your `MsgBox` call, summarized in the following table:

Prompt	Part1	Part2	"Solution" displayed?	Part3
Part1 @Part2 @Part3	Bold	Normal	Yes	Normal
Part1 @Part2 @	Bold	Normal	No	
@Part2 @Part3		Normal	Yes	Normal

If you are creative, use this mechanism to help highlight your error messages, or use it like Microsoft does, to format "*Problem:Solution*" message boxes.



This demo shows two new features of MsgBox: adding a Help button, and using the "@" symbol to format the text. (It's in TestMsgBox in basVBASamples.)

Working with Forms and Reports

Visual Basic also adds some functionality to forms and reports that include their own modules. You can now create user-defined properties, create multiple instances of forms or reports at runtime, and cause reports and forms to open automatically, hidden, when you set or retrieve a property of that object.

User-Defined Properties

As the previous section points out, you can refer to Public variables in form and report modules from outside those modules. Effectively, this allows you to create new properties on forms and reports: because you read and write these public variables, using a syntax that exactly matches the syntax you use to work with a built-in property of a form or report.

Sometimes, though, you want to set a property of a form or report that requires some reaction from the object once you set the property. For instances like these, Visual Basic provides the Property Let/Get/Set mechanism. These special constructs allow you to write procedures that control what happens when a caller sets the value of, attempts to retrieve the value of, or, in the case of Object properties, attempts to Set an object variable to refer to the property.

For example, frmSampleForm supports the user-defined TextColor property. Normally, forms do not have a TextColor property, but you would like to be able to set the text color for all the form's text boxes in one property setting. Visual Basic makes this simple. You can assign and retrieve the value of this property with procedures like the following:

```
Property Let BackColor(lngColor As Long)
    ' Set all the text boxes to have the chosen color
    ' as their background color.
    Dim ctl As Control
    For Each ctl In Me.Controls
        If ctl.ControlType = acTextBox Then
            ctl.BackColor = lngColor
        End If
    Next ctl
End Property

Property Get BackColor() As Long
    ' Find the first textbox and retrieve its
```

```
' background color.  
Dim ctl As Control  
TextColor = 0  
For Each ctl In Me.Controls  
    If ctl.ControlType = acTextBox Then  
        BackColor = ctl.ForeColor  
        Exit For  
    End If  
Next ctl  
End Property
```

To set and retrieve the property, use the normal syntax:

```
Forms!frmSampleForm.BackColor = 255  
Debug.Print Forms!frmSampleForm.BackColor
```



To try out form properties, open `frmSampleForm` and set its `BackColor` property. (Of course, forms don't have a `BackColor` property: it's user-defined!)

Multiple Instances of Forms/Reports

Access 95 allows you to create multiple instances of forms and reports, each with their own current row and their own settings. Using the `New` keyword, you can create a new instance of any existing form or report:

```
Set frm = New Form_frmCustomers
```

The example form, `frmCustomers`, shows this technology in use (see Figure 50).

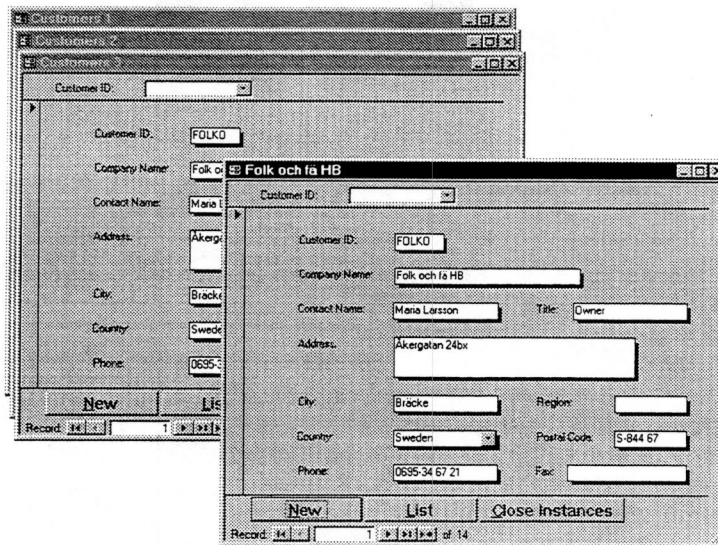


Figure 50. You can create multiple instances of forms and reports in Access 95.

Here's the code frmCustomers uses to keep track of the instances:

```
Dim colForms As New Collection
Dim mintI As Integer

Sub NewCustomerForm(ByVal frmParent As Form)
    Dim frm As Form

    Set frm = New Form_frmCustomers
    mintI = mintI + 1
    ' The Key value must be a string, so tack on a
    ' null string to force the conversion. You'll
    ' use the hWnd later when you try and
    ' remove the window from the collection of windows.
    colForms.Add Item:=frm, Key:=frm.hwnd & ""
    frm.Caption = "Customers " & mintI
    DoCmd.MoveSize (mintI + 1) * 80, (mintI + 1) * 350

    frm.Visible = True
End Sub

Sub ListInstances()
    Dim frm As Form
    For Each frm In colForms
        Debug.Print frm.Caption
    
```



```
Next frm
End Sub

Sub CloseInstances()
    Dim intI As Integer
    Dim frm As Form
    ' The user may have closed some or all of the
    ' forms by hand. Skip any errors that
    ' occur because the collection count
    ' doesn't match reality.
    For intI = 1 To colForms.Count
        ' Either you need to step backwards,
        ' or just always close the first element
        ' in the collection.
        ' Because Access will close the forms when
        ' the variable pointing to them goes
        ' out of scope, calling the Remove method will
        ' close all the forms for you.
        colForms.Remove 1
    Next intI
    mintI = 0
End Sub

Sub RemoveInstance(frm As Form)
    ' Each form calls this code when it closes itself.
    ' That works great, and is hooked to the main
    ' form's Close event.
    On Error Resume Next
    colForms.Remove frm.hwnd & ""
End Sub
```

Some important points to keep in mind when using these new multiple-instance forms (all points pertain to reports, as well):

- All the form instances have the same name, and therefore cannot be referenced in the Forms collection by name. They do each have a separate index when you reference items in the collection by number.
- To refer to the forms once you create them, you need to store the reference in some safe place. The example code (in `basFormInstance`) uses a collection of form objects.
- Each form can have a different current row, and different properties.

- You cannot save changes to non-default forms.
- Closing the default instance does not close copies of the form.
- If you store the form references in a collection, react to the form's Close event and remove the reference from the collection. The example code uses the form's hWnd (guaranteed to be unique) to remove itself from the collection.



This demo shows how multiple instances of forms works. The sample form, frmCustomers, uses the code shown above to create new instances of itself, adding forms' window handles to a collection of handles.

Form/Report Auto-Open

In Access 2.0, if you wanted to open a form and change some of its properties before the user saw the form, you had to open it twice, first as a hidden form, and then as a normal form. Visual Basic allows you to refer to a closed form, and set its properties. The first time you refer to the form, Access opens it for you, hidden. Then, when you want to see the object, set its Visible property to be True. For example:

```
Form_frmCustomers.Caption = "This is a test"  
Form_frmCustomers.Visible = True
```

You can even retrieve information about closed forms and reports this way:

```
Debug.Print Form_frmCustomers.Controls.Count
```

As long as you remember that the object has been opened, but is hidden, this is a useful technique. Because there is no way to close the object without first making it visible, use the technique with care.



This demonstration works with form properties for a form that is not yet opened. When you attempt to set a property for a form, Access opens it, hidden, if it is not already open. It is up to you to set its Visible property to True, if you want to see it.

Windows API Issues

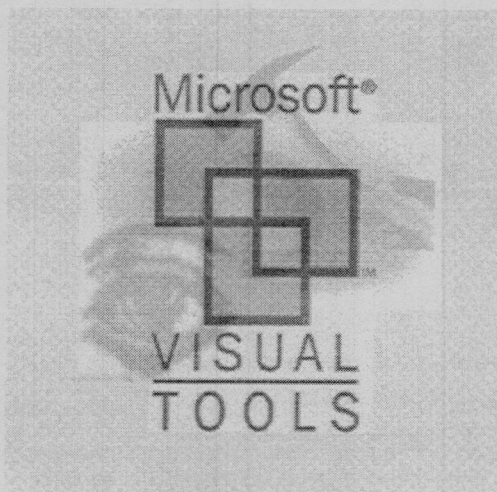
Though not directly associated with Visual Basic, calling the Windows API from Access 95 is a difficult issue. Because you cannot call 16-bit DLLs from a 32-bit application, and because the 32-bit versions of all Windows API functions have new names and often new parameters and return values, every single Windows API call and declaration needs to be modified.

The major points:

- Every library name is now different (usually just a "32" tacked on the end).
- Almost every handle object has been changed from a 16-bit value to a 32-bit value (integer to long).
- Windows NT (and one day, Windows 95) fully supports both ANSI and Unicode character sets. You have to tell Windows which version of a function (ANSI or Unicode) you are calling, because the API includes both for any functions that involve text. To do this, you must provide an alias for each function, telling Windows that you actually need to call a function like `GetProfileStringA` or `GetProfileStringW` when you call `GetProfileString` in your code.
- Most functions that used to return integers now return longs.
- Many functions that were in common use have been superseded or no longer exist.

The Access Developer's Toolkit (and VB 4.0) ship with a tool to help you look up the correct declarations for your API calls. In addition, the Microsoft Developer Network CD includes descriptions of every API call, to help you find the right one. Needless to say, this issue is going to be the largest single issue involved in converting from Access 2.0 to Access 95.

.....



Microsoft® Developer
Seminar-in-a-Box Series

Microsoft Access 95
Demonstration Script

Introduction

This demonstration script was developed for the “*Building Solutions with Microsoft Visual Tools*” Seminar. Files mentioned in this document can be found on the CD-ROM distributed with the seminar.

It is essential that the person doing the demonstration be familiar with Microsoft Access 7.0 for Windows 95 and the development disciplines associated with creating multi-user and client/server databases.

Before doing this demonstration, be sure to read the associated Whitepapers located on the seminar CD-ROM .

Setup Instructions

This demonstration assumes that you are running either Microsoft Windows NT 3.51 or higher (Workstation or Server) or Microsoft Windows 95 and have installed Microsoft Access 7.0 for Windows 95 with all of the options. You will need some of the sample files from Microsoft Access in order to complete some demo sections.

System Requirements

Specification	Minimum Requirements
CPU	486 DX2/33MHz
RAM	16 M
Free Disk Space (for the demo files)	N/A
Operating System	Windows NT 3.51 (Workstation. or Server) or Windows 95
Additional Software	Microsoft Access 7.0 for Windows 95 (complete install) Microsoft Excel 7.0 for Windows 95 (complete install) Access Developers Toolkit for Windows 95

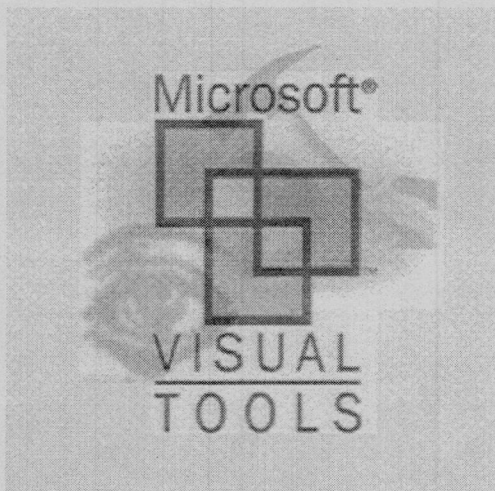
Demonstration Instructions

All demonstrations are documented within the Microsoft Access 7.0 for Windows 95 Instructor’s manual. Demonstrations are indicated by the following icon:

To the right of the icon are instructions on how to perform the suggested demonstration. The instructions assume that the instructor has enough knowledge to interpret the instructions and create any necessary demonstration files necessary to carry the instructions out. If the instructions in the Instructor’s Guide aren’t readily visible, they are hidden text. From within Word 7.0 for Windows 95, go to Tools|Options and indicate under non-printing characters that you want to see hidden text. If you are going to print the Instructor’s Guide, you may choose to change all of the hidden text to a different style.

Note that the Instructor’s Guide and the Student Guide are identical except for the hidden text. Note that the page numbers will be different between the two books if you make the hidden text visible in the Instructor’s Guide and print it.

.....

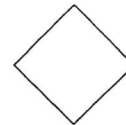


***Microsoft*® Developer
Seminar-in-a-Box Series**

**Microsoft Access 95
Seminar Slides**

Legal Disclaimer and copyright slide

**Please remove this slide prior
to presenting, but leave it in
for any electronic or
material photocopying or
distribution**



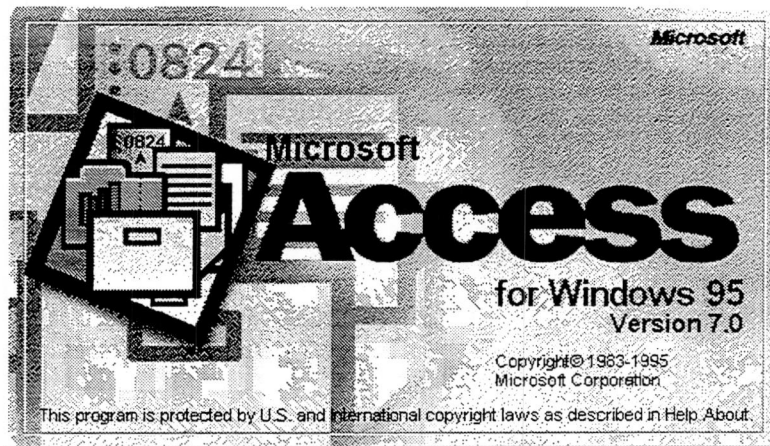
Microsoft, Windows, and Win32, are registered trademarks and Microsoft Access is a trademark of Microsoft Corporation. All other trademarks, marked and not marked, are property of their respective owners.

This document is provided for informational purposes only. The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to change in market conditions, it should not be interpreted to be a commitment on the part of Microsoft and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

INFORMATION PROVIDED IN THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND FREEDOM FROM INFRINGEMENT. The user assumes the entire risk as to the accuracy and the use of this document. This document may be copied and distributed subject to the following conditions: 1) All text must be copied without modification (except foreign language translation) and all pages must be included; 2) All copies must contain Microsoft's and Application Developer Training Company's copyright notice and any other notices provided therein; and 3) **This document may not be distributed within the boundaries of Canada or the United States of America.**

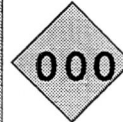
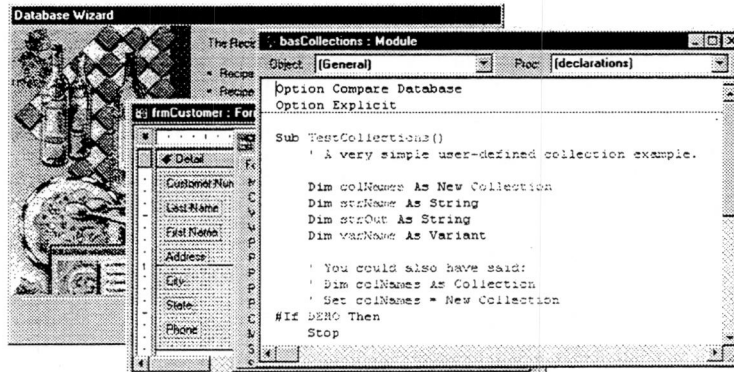
Copyright © 1996 Application Developers Training Company. All Rights Reserved.

What's New in Microsoft Access 95



Agenda for the Day

- ◆ What's New for Everyone?
- ◆ What's New for Developers?
- ◆ What's New for Coders?

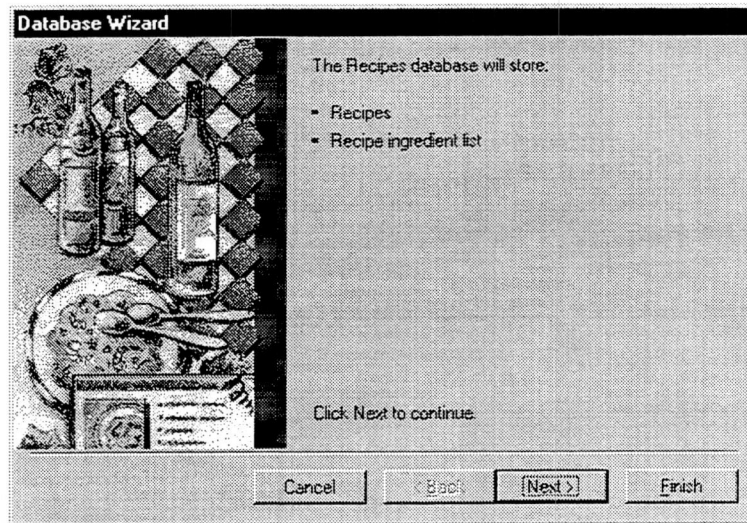


Instructor Name Here

◆ **Instructor Notes Here**



What's New for Everyone?



Agenda

- ◆ **Database Improvements**
- ◆ **Table Improvements**
- ◆ **Query Improvements**
- ◆ **Form Improvements**
- ◆ **Report Improvements**
- ◆ **Macro Improvements**

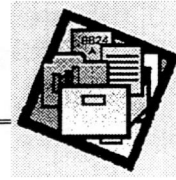


Agenda

- ◆ Database Improvements
- ◆ Table Improvements
- ◆ Query Improvements
- ◆ Form Improvements
- ◆ Report Improvements
- ◆ Macro Improvements



The Database Wizard



- ◆ Builds 22 common databases
- ◆ Allows you to customize
 - Optional Fields
 - Styles for Forms and Reports
- ◆ Builds an entire application
 - Tables
 - Queries
 - Forms
 - Reports



The Answer Wizard

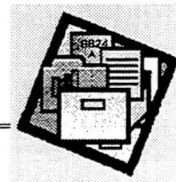


- ◆ **“Fuzzy” Search Tool for Help Files**
- ◆ **Tries to locate topics by looking at keywords and synonyms**
- ◆ **The best way to search when you’re not sure what technical term the help file uses for something**





The Database Explorer



- ◆ Four views of your database
- ◆ Easy sorting of objects by name, date, or description
- ◆ Easy renaming of objects
- ◆ Shortcut menus for objects
- ◆ Object Property Sheets



Agenda

- ◆ **Database Improvements**
- ◆ **Table Improvements**
- ◆ **Query Improvements**
- ◆ **Form Improvements**
- ◆ **Report Improvements**
- ◆ **Macro Improvements**



Agenda

- ◆ Database Improvements
- ◆ Table Improvements
- ◆ Query Improvements
- ◆ Form Improvements
- ◆ Report Improvements
- ◆ Macro Improvements



Table By Data

- ◆ You can create a table just by typing in a datasheet!
- ◆ Right-click performs column operations:
 - Rename
 - Delete
 - Insert
- ◆ Access 95 picks field datatypes based on the data you enter



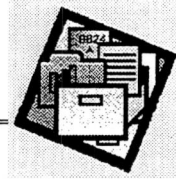
Table Lookups

- ◆ Use the Lookup Wizard to build intelligent foreign key fields
- ◆ Lookup Properties in table design view controls how these fields are displayed
- ◆ This works in all datasheets and is inherited by forms and reports





More Table Improvements



- ◆ **Designing tables can be launched from the Relationships Window**
- ◆ **Excel and text files can be linked as well as imported**



Agenda

- ◆ Database Improvements
- ◆ Table Improvements
- ◆ Query Improvements
- ◆ Form Improvements
- ◆ Report Improvements
- ◆ Macro Improvements



Agenda

- ◆ Database Improvements
- ◆ Table Improvements
- ◆ Query Improvements
- ◆ Form Improvements
- ◆ Report Improvements
- ◆ Macro Improvements



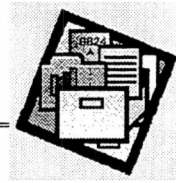
Simple Query Wizard

- ◆ **A *great* tool for building**
 - Single-table select queries
 - Multi-table select queries
 - Totals queries
- ◆ **AutoJoins**
 - The Simple Query Wizard automatically includes linking tables when they're needed





Formatting in Queries



- ◆ **Datasheet Formatting Toolbar**
 - Foreground and background color
 - Font formatting
 - Gridlines and special effects
- ◆ **Cell Effects Dialog**
- ◆ **QuickSort in Queries**



Agenda

- ◆ **Database Improvements**
- ◆ **Table Improvements**
- ◆ **Query Improvements**
- ◆ **Form Improvements**
- ◆ **Report Improvements**
- ◆ **Macro Improvements**



Agenda

- ◆ **Database Improvements**
- ◆ **Table Improvements**
- ◆ **Query Improvements**
- ◆ **Form Improvements**
- ◆ **Report Improvements**
- ◆ **Macro Improvements**



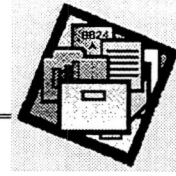
Form Wizards

- ◆ **AutoForm**
 - Columnar
 - Tabular
 - Datasheet
- ◆ **Form Wizard**
 - Single or Multi-Table
 - SubForms or Linked Forms
 - Form Styles
- ◆ **Background Pictures**





New Filtering Methods



Filter by Selection

- Quickly find records that match the current record



Filter by Form

- Create complex filtering queries with a simple point-and-click interface



Agenda

- ◆ **Database Improvements**
- ◆ **Table Improvements**
- ◆ **Query Improvements**
- ◆ **Form Improvements**
- ◆ **Report Improvements**
- ◆ **Macro Improvements**



Agenda

- ◆ **Database Improvements**
- ◆ **Table Improvements**
- ◆ **Query Improvements**
- ◆ **Form Improvements**
- ◆ **Report Improvements**
- ◆ **Macro Improvements**



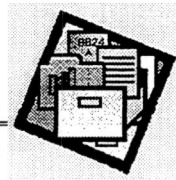
Report Wizards

- ◆ **AutoReport**
 - **Columnar**
 - **Tabular**
- ◆ **Report Wizards**
 - **Single or Multiple Tables**
 - **Visual Report Grouping**
 - **Report Styles**





Other Report Improvements



- ◆ **Insert Page**
- ◆ **Flexible Zoom**
- ◆ **Multi-page Print Preview**
- ◆ **Export to Word or Excel with subreports**



Agenda

- ◆ Database Improvements
- ◆ Table Improvements
- ◆ Query Improvements
- ◆ Form Improvements
- ◆ Report Improvements
- ◆ Macro Improvements



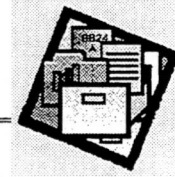
Agenda

- ◆ Database Improvements
- ◆ Table Improvements
- ◆ Query Improvements
- ◆ Form Improvements
- ◆ Report Improvements
- ◆ Macro Improvements





What's New in Macros



- ◆ **Save Action**
- ◆ **Macro to Module Converter**
 - Takes any macro and converts it to Visual Basic
 - The code may need cleaning up and improving, but it should run without changes



Questions?



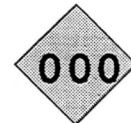
Agenda

- ◆ Office Compatibility
- ◆ OLE



Agenda

- ◆ Office Compatibility
- ◆ OLE

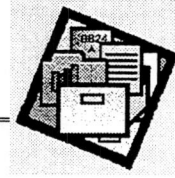


Opening Files

- ◆ **Enhanced File Open Dialog**
 - Subfolder views
 - File search capabilities
- ◆ **Database Properties**
- ◆ **Advanced Searching**



Text Import Wizard

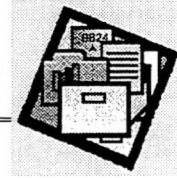


- ◆ For both delimited and fixed-width text files
- ◆ Same design and interface as the Excel Text Import Wizard





Office Compatibility



- ◆ **Spelling tools**
 - Spell Checking
 - AutoCorrect
- ◆ **Tear-off Palettes**
- ◆ **Toolbar Button Faces**
 - Button face editor
 - Past to button



Agenda

- ◆ Office Compatibility
- ◆ OLE

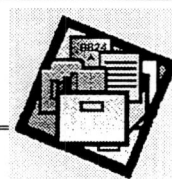


Agenda

- ◆ **Office Compatibility**
- ◆ **OLE**



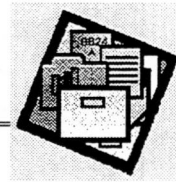
OLE Drag and Drop



- ◆ You can drop Access objects almost anywhere
 - On Word to create a table
 - On Excel to create a worksheet
 - On another copy of Access to copy the object



Pivot Table Wizard

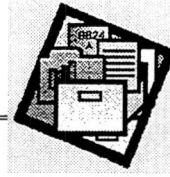


- ◆ A tool to create Excel Pivot tables on Access forms
- ◆ A cooperative Wizard, running partly in Access and partly in Excel

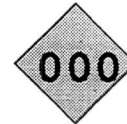




Windows 95 Shortcuts



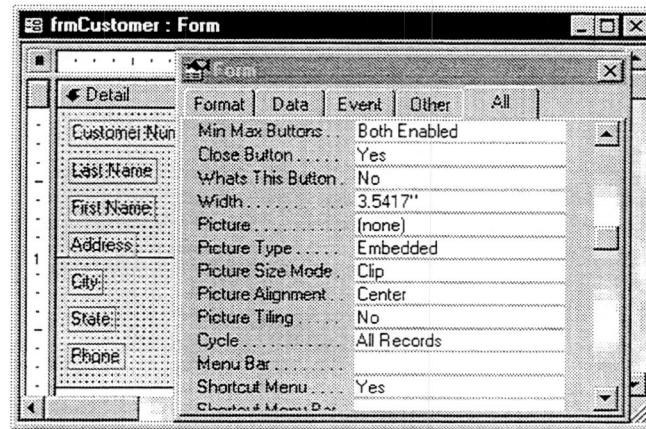
- ◆ Drag objects from Access databases to the Windows 95 desktop to create shortcuts
- ◆ You can insert these shortcuts in other applications
- ◆ Double-click the shortcut to open Access, load the database, and display the object



Questions?



What's New for Developers?



000

Agenda

- ◆ **Database Tools and Features**
- ◆ **Solution Distribution**
- ◆ **Form Design Improvements**
- ◆ **OLE Support**

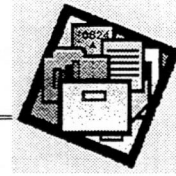


Agenda

- ◆ Database Tools and Features
- ◆ Solution Distribution
- ◆ Form Design Improvements
- ◆ OLE Support



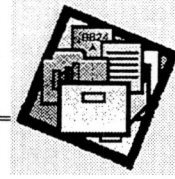
Database Options



- ◆ **Tools|Options replaces View|Options**
- ◆ **New Options**
 - **Filter by Form options under Edit/Find**
 - **Look and feel options under Datasheets**
 - **AutoIndex and data type defaults under Table/Query**
 - **Code and editor options under Module**



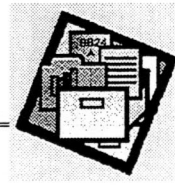
The Table Analyzer



- ◆ **Helps Normalize your databases**
- ◆ **Can split tables containing information on multiple entities into their component parts**
- ◆ **Helps correct typographical errors in imported information**
- ◆ **Supports automatic or manual operation**



The Performance Analyzer



- ◆ Reviews database objects for performance issues
- ◆ Will fix objects that have performance problems on command
- ◆ Can make a dramatic difference in database speed

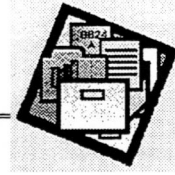


Replication

- ◆ Access 95 supports replication, which can keep multiple copies of a database synchronized
 - Copies for satellite offices
 - Mobile copies
 - Warm backups
 - Load balancing
 - Distributing updates



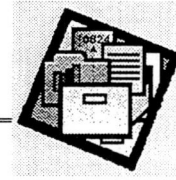
The Replication Process



- ◆ Turn your database into a design master
- ◆ Make replicas from the design master
- ◆ Work with the replicas
- ◆ Synchronize between replicas
- ◆ Resolve conflicts



Synchronization



- ◆ Any two replicas with the same ancestor can be synchronized
- ◆ Access automatically resolves conflicting changes to data
- ◆ The losing copy gets a chance to reverse the automatic decision with the Conflict Resolution Wizard



Replication Notes

- ◆ All replication activities can be performed from any program that can use the Jet engine, including Access 95, Visual Basic 4, and Visual C++
- ◆ Briefcase replication can easily synchronize replicas between home and office or desktop and laptop
- ◆ Replication Manager provides advanced management tools for networks



Questions?



Agenda

- ◆ **Database Tools and Features**
- ◆ **Solution Distribution**
- ◆ **Form Design Improvements**
- ◆ **OLE Support**



Agenda

- ◆ **Database Tools and Features**
- ◆ **Solution Distribution**
- ◆ **Form Design Improvements**
- ◆ **OLE Support**

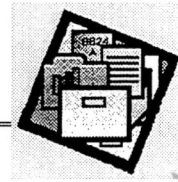


Startup Properties

- ◆ **Application Title**
- ◆ **Display Form**
- ◆ **Application Icon**
- ◆ **Menu Bar and Shortcut Menu Bar**
- ◆ **Disable various features**



Database Splitter Wizard



- ◆ Removes tables to a separate database
- ◆ Links the tables back to the original database
- ◆ Ideal for shared applications on a network

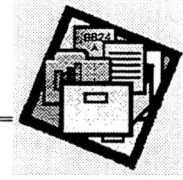


The Access Developer's Toolkit

- ◆ **Runtime License & Setup Wizard**
- ◆ **Language Reference & DAO Reference**
- ◆ **Custom Controls**
- ◆ **Win32 API Viewer**
- ◆ **Replication Manager & Transporter**
- ◆ **Microsoft Help Workshop**



Setup Wizard



- ◆ Prepares your application for distribution
- ◆ Can create custom shortcuts and registry entries
- ◆ Handles both diskette and network setup



Questions?



Agenda

- ◆ Database Tools and Features
- ◆ Solution Distribution
- ◆ Form Design Improvements
- ◆ OLE Support



Agenda

- ◆ **Database Tools and Features**
- ◆ **Solution Distribution**
- ◆ **Form Design Improvements**
- ◆ **OLE Support**



Form Design Agenda

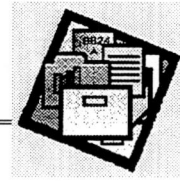
- ◆ **Creating Forms**
- ◆ **New Control Properties**
- ◆ **Working with Controls**
- ◆ **Working with Combo and List Boxes**
- ◆ **Working with Subforms and Datasheets**
- ◆ **Advanced Form Properties and Events**



Creating Forms



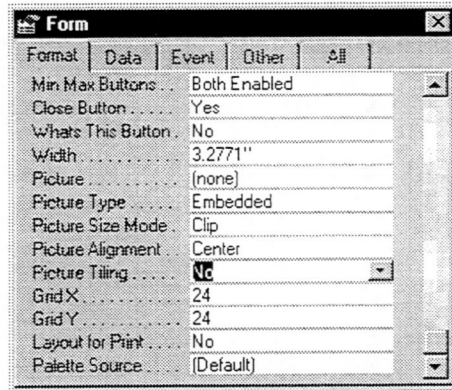
Forms and Filtered Data



- ◆ Create forms based on filtered datasheets
- ◆ Filter and sort saved with form
- ◆ Sort active when you open form
- ◆ Filter only active when applied



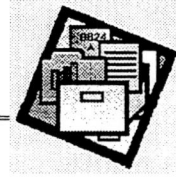
Tabbed Property Sheets



Property	Value
Min Max Buttons	Both Enabled
Close Button	Yes
Whats This Button	No
Width	3.2771"
Picture	(none)
Picture Type	Embedded
Picture Size Mode	Clip
Picture Alignment	Center
Picture Tiling	No
Grid X	24
Grid Y	24
Layout for Print	No
Palette Source	(Default)



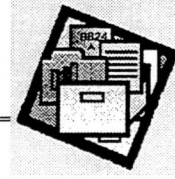
Cycled Properties



- ◆ Double-click on a property on a property sheet to switch between property values
- ◆ Works with any property that has a fixed list of possible values



AutoFormat



- ◆ Use a saved format
- ◆ Create your own named formats
- ◆ Create a "look" for your company's forms and reports
 - Make it available to all your developers

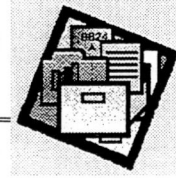


Default Control Types

- ◆ When you create tables, you can select default control types for fields
- ◆ Specify lookup information for combo/list boxes
- ◆ Wizards create forms and reports with the correct controls



Control Morphing

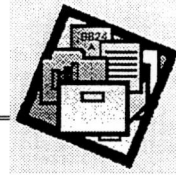


- ◆ Convert a control to another control type
- ◆ Access preserves appropriate properties





What's This Button



- ◆ Add a *What's This* button to your form, like built-in dialogs have
- ◆ Allows context-sensitive help on any item
- ◆ You supply the `HelpFileName`
- ◆ You supply `HelpContextID` for controls
- ◆ Must remove Min/Max buttons



New Control Properties



Control Tips

- ◆ **Add control tips to any control**
 - **Usually limiting this to command buttons works best**
- ◆ **You just supply the ControlTipText**



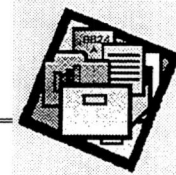
Shortcut Menus

- ◆ **Add context-sensitive menus**
 - **Popup in reaction to right mouse-click**
- ◆ **Use the menu builder to create a shortcut menu**





Allow AutoCorrect



- ◆ **AutoCorrect is great**
 - Sometimes you want to turn it off
 - Proper names shouldn't be spell checked
- ◆ **Use the AllowAutoCorrect property**
 - Set to Yes/No to allow/disallow for any field



Working with Controls



Using the Image Control

- ◆ **Unbound OLE object still useful**
 - It requires more overhead
 - It allows editing
- ◆ **If you don't need to supply editing capabilities**
 - Use the new Image Control
 - It's faster with less overhead

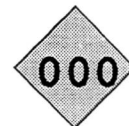
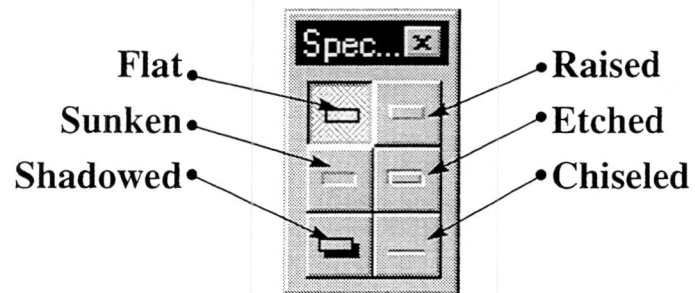


The Format Painter

- ◆ Format Painter copies formatting from the selected control
- ◆ Applies it to other control(s)
- ◆ Double-click to make persistent

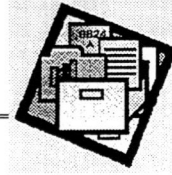


Control Special Effects





Triple-State Controls



- ◆ **Allow three states:**
 - Yes
 - No
 - Null ("I don't know")
- ◆ **For ungrouped**
 - Option Buttons
 - Check boxes
 - Toggle Buttons



Working with Combo and List Boxes



Improved LimitToList

- ◆ Now it's possible to enter a value
 - and then delete it
 - Even if LimitToList is set to True
- ◆ In Access 2, this required some extra code



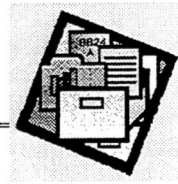
Multi-Select List Boxes

- ◆ **Three options for Multi-Select:**
 - **None (the old style)**
 - **Simple**
 - **Click on items to select and deselect**
 - **Extended**
 - **Click on item**
 - **Ctrl+Click selects discontiguous**
 - **Click/Shift+Click for contiguous**





Multi-Select List Boxes

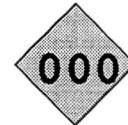


◆ How to retrieve the selected list?

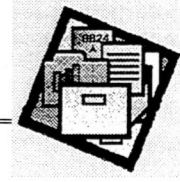
- ItemsSelected collection
 - Retrieve collection of selected indexes
- Selected property
 - Array of items, selected or not?



Working with Subforms and Datasheets



Subform Wizard



- ◆ Creating subforms has never been easier!
- ◆ Even if the underlying multi-table query hasn't yet been built, the wizard can do it



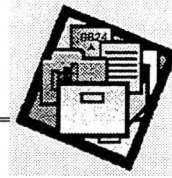
Subform Linking Wizard

- ◆ Use the subform linking wizard to help
 - Build the LinkMasterFields/LinkChildFields
 - Link one or two fields from master and child forms





Work with Selected Rows



- ◆ Finally, you have a way to know which rows/columns the user has selected
- ◆ SelTop, SelLeft, SelWidth, SelHeight properties give you the information



Advanced Form Properties and Events



Using the Filter/Sort Properties

- ◆ **Filter** property contains the filter
- ◆ **FilterOn** property indicates whether the filter is active
- ◆ **OrderBy** property contains the sort (added to the recordset's Order By)
- ◆ **OrderByOn** property indicates whether the sort is active
- ◆ For forms, **OrderByOn** defaults to True:
FilterOn defaults to False



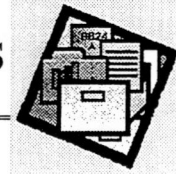
Cycle Property

- ◆ **Cycle through controls on your form**
- ◆ **None**
 - the old way, moves you to next/previous row
- ◆ **Current Record**
 - loops through controls on the current row
- ◆ **Current Page**
 - loops through controls on the current page
(for multi-paged forms)





KeyPreview and Key Events



- ◆ Use KeyPreview property to enable form to react to key events before controls
- ◆ Access sends all keystrokes to the form
- ◆ Form processes keystrokes
- ◆ Form can "swallow" keystrokes, or let them pass on through
- ◆ In test case, cause form to disregard PgUp and PgDn keys.

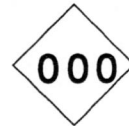


Questions?



Agenda

- ◆ **Database Tools and Features**
- ◆ **Solution Distribution**
- ◆ **Form Design Improvements**
- ◆ **OLE Support**



Agenda

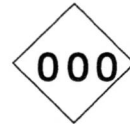
- ◆ Database Tools and Features
- ◆ Solution Distribution
- ◆ Form Design Improvements
- ◆ OLE Support



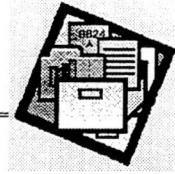
OLE Controls

◆ **Benefits of using OLE controls:**

- Tested, debugged, ready for use
- Tightly integrated into apps, and can be data-bound
- Shared between application environments



Bundled OLE Control



- ◆ Access 95 ships with Calendar OLE control
- ◆ Simple-to-use control
 - works like any calendar
 - provides methods, events and properties
- ◆ Custom controls now display their properties on the built-in property sheet

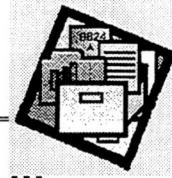


OLE Controls in the ADT

- ◆ **Win95 Common Controls**
- ◆ **Common Dialogs**
- ◆ **Rich Text Box**
- ◆ **Slider**
- ◆ **Toolbar**
- ◆ **Data Outline Control**



A Caveat



- ◆ Not all Visual Basic 4 controls work in Access 95
- ◆ Two types that won't work:
 - Controls that contain other controls
 - Controls that are bound to complex data sources (whole rows, rather than a column)
 - Bound grid controls, for example
- ◆ Ask the vendor, to ensure compability



References Dialog

- ◆ Allows you to declare references to external libraries and databases
- ◆ Provides programmatic access to constants, objects, and methods
- ◆ Makes using the Object Browser possible

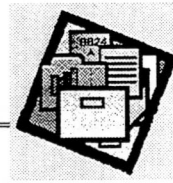


Object Browser

- ◆ View and learn about OLE objects
- ◆ Allows you to paste code directly into your application
- ◆ Investigate object models for any OLE server (including Access itself)



Object Browser



- ◆ Using the object browser, you can:
 - Look up the object in online help
 - Dig deeper into the object hierarchy
 - Paste the syntax for the object into your code
 - Choose another object to browse



OLE Automation Server

- ◆ Access has been an OLE Automation controller since Access 2
- ◆ Now it's a Server, as well
- ◆ You can automate Access actions from any application that can act as an OLE Automation controller
 - Excel
 - Project
 - Visual Basic 4



Controlling Access

- ◆ Anything you can do programmatically in Access you can do from outside Access, as well
- ◆ DoCmd object: macro actions
- ◆ Application object: UI objects
- ◆ DAO: data access objects



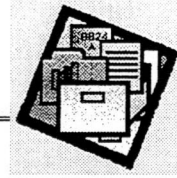
One more thing...

- ◆ Is the user in control, or is OLE automation?
- ◆ The Application.UserControl property tells you what's going on
- ◆ Application.Visible property
 - Read-only if user in control
 - Read/write if OLE in control





Common Usage



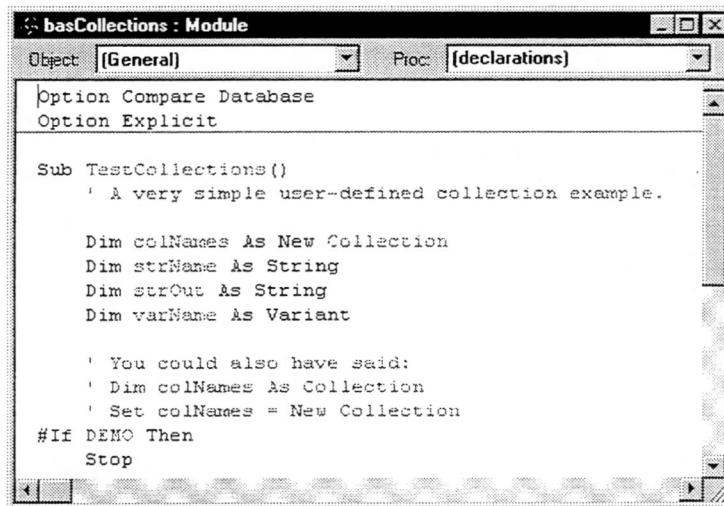
- ◆ **Running reports in Access, from Excel**
- ◆ **The Access report writer is the best**
- ◆ **Can link Excel data directly to Access**
- ◆ **OLE Automation allows you to run reports from an Excel application**



Questions?



What's New for Coders?



```
basCollections : Module
Object: (General) Proc: (declarations)
Option Compare Database
Option Explicit

Sub TestCollections()
    ' A very simple user-defined collection example.

    Dim colNames As New Collection
    Dim strName As String
    Dim strOut As String
    Dim varName As Variant

    ' You could also have said:
    ' Dim colNames As Collection
    ' Set colNames = New Collection
    #If DEMO Then
        Stop
    End Sub
```

000

Agenda

- ◆ **VBA IDE**
- ◆ **Environment changes**
- ◆ **Language changes**
- ◆ **Working with Forms and Reports**
- ◆ **Windows API issues**



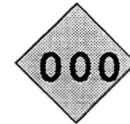
Agenda

- ◆ **VBA IDE**
- ◆ **Environment changes**
- ◆ **Language changes**
- ◆ **Working with Forms and Reports**
- ◆ **Windows API issues**



What's VBA?

- ◆ **Visual Basic for Applications**
- ◆ **Shared with:**
 - **Project**
 - **Visual Basic 4.0**
 - **Excel for Windows 95**
 - **Word for Windows 95?**
 - **Not this time!**
- ◆ **"Visual Basic" refers to the language**
 - **not the product Visual Basic 4.0**



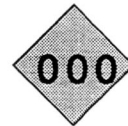
Color Coded Editing

- ◆ **Finally!**
- ◆ **Use Tools|Options|Module dialog to control**
- ◆ **Turn off Auto Syntax Checking**



Customizable Font

- ◆ Save your eyes!
- ◆ Great for presentations



Line Continuation

- ◆ No more endless lines
- ◆ No more breaking up long SQL strings and concatenating the pieces
- ◆ Four characters:
 - Space, Underscore, CR/LF

```
DoCmd.OpenForm _  
  FormName:="frmSampleForm", _  
  WindowMode:=acDialog, _  
  OpenArgs:="Smith"
```

000

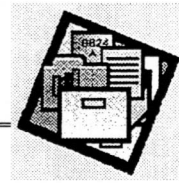
Line Continuation, cont'd

- ◆ Break almost anywhere:

- Not in quoted strings (but you can break a long string in two pieces concatenated with the & operator)
- Not in keyword or variable



Full Module View



- ◆ Allows you to show all procedures in a module in one long stream
- ◆ Use Tools|Options|Module|Full Module View to set
- ◆ Show or hide the procedure separator with Tools|Options|Module|Procedure Separator



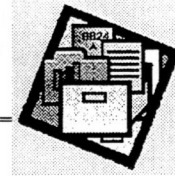
Watch Points

- ◆ Allows you to watch any expression's value
- ◆ No need to use `Debug.Print` or `?` in Debug Window
- ◆ Use `Tools|Instant Watch` or `Tools|Add Watch` to add a watch point
- ◆ Shows up in Debug Window





Conditional Breakpoints



- ◆ Rather than setting normal breakpoints
- ◆ Break only if a certain condition is true
- ◆ Or if an expression changes its value
- ◆ Useful to allow you to break when the condition you care about becomes true



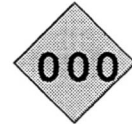
Agenda

- ◆ VBA IDE
- ◆ Environment changes
- ◆ Language changes
- ◆ Working with Forms and Reports
- ◆ Windows API issues



Agenda

- ◆ VBA IDE
- ◆ Environment changes
- ◆ Language changes
- ◆ Working with Forms and Reports
- ◆ Windows API issues



The DoCmd Object

- ◆ To support OLE Automation, DoCmd has to be an object
- ◆ Therefore, all macro actions are now methods of the DoCmd Object
- ◆ Access 95 does the conversion for you

```
DoCmd.OpenForm "frmSampleForm"
```



Conditional Compilation

- ◆ Control whether or not to include certain sections of code
- ◆ #If, #End If, #Else, #ElseIf, #Const

```
' Reset this value when released
#Const DEBUG = True
#If DEBUG Then
    ' Use this code for debugging
#Else
    ' Use this code for normal use
#End If
```

000



Search All Modules

- ◆ **No more opening forms and reports by hand**
- ◆ **Find can now search through all modules**
- ◆ **Includes form and report modules**
- ◆ **Choices:**
 - **Current Procedure**
 - **Current Module**
 - **Current Database**
 - **Selected Text**



Agenda

- ◆ **VBA IDE**
- ◆ **Environment changes**
- ◆ **Language changes**
- ◆ **Working with Forms and Reports**
- ◆ **Windows API issues**



Agenda

- ◆ VBA IDE
- ◆ Environment changes
- ◆ Language changes
- ◆ Working with Forms and Reports
- ◆ Windows API issues



Built-In Constants

- ◆ Dir, GetAttr, SetAttr
- ◆ StrConv
- ◆ Shell
- ◆ VarType
- ◆ FirstWeekOfYear, FirstDayOfWeek



MsgBox Constants are in there!

```
MsgBox "Error!", _  
vbCritical + vbAbortRetryIgnore + _  
vbDefaultButton1
```

000

Miscellaneous Constants

- ◆ **vbCrLf: Chr\$(13) & Chr\$(10)**
- ◆ **vbNullChar: Chr\$(0)**
- ◆ **vbTab: Chr\$(9)**
- ◆ **vbBack: Chr\$(8)**
- ◆ **among others...**

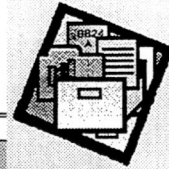


For Each...Next

- ◆ Loop through all the elements of a collection or array
- ◆ No index needed
- ◆ Don't need to know the bounds
- ◆ Access SET's the loop variable equal to each object in the collection, in turn



For Each...Next



```
'cmdEnumerate_Click in frmSampleForm  
For intI = Me.Controls.Count - 1  
    Set ctl = Me.Controls(intI)  
    Debug.Print ctl.Name, ctl.ControlType  
Next intI
```

Old way: ↑

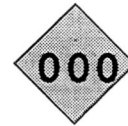
New way: ↓

```
For Each ctl In Me.Controls  
    Debug.Print ctl.Name, _  
        ctl.ControlType  
Next ctl
```

000

For Each...Next Fine Points

- ◆ "Loop" variable can be object, or variant
- ◆ For Each...Next works with any collection
- ◆ Can use For Each...Next to retrieve data from arrays
 - but you cannot set data in arrays because of the internal data structures



For Each with Arrays

- ◆ Allows you to retrieve (not set) each element of an array
- ◆ Loop variable must be a variant
- ◆ Never need to worry about UBound and LBound again!

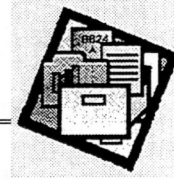


With...End With

- ◆ **Work with multiple properties, methods, objects of a specific object**
- ◆ **Saves typing, speeds up code**



With...End With



```
Forms!frmSampleForm.Width = 1400  
Forms!frmSampleForm.Caption = "This is a test"  
Forms!frmSampleForm!cmdEnumerate.Caption = _  
    "Enumerate Controls"
```

Old way: ↑

New way: ↓

```
With Forms!frmSampleForm  
    .Width = 1400  
    .Caption = "This is a test"  
    !cmdEnumerate.Caption = _  
        "Enumerate Controls"  
End With
```

000

Named Parameters

- ◆ No more counting commas!
- ◆ Once you use named parameters, all the rest in the call must be named
- ◆ Order doesn't matter for named parameters



Named Parameters

```
DoCmd.OpenForm "frmSampleForm", , , , , _  
acDialog, "Smith"
```

Old way: ↑

New way: ↓

```
DoCmd.OpenForm _  
FormName:="frmSampleForm", _  
OpenArgs:="Smith", WindowMode:=acDialog
```

000

Named Parameters (still more!)

- ◆ They work great for user-defined procedures, too

```
' In basVEASamples  
Sub HandleItems(frm As Form, _  
    ForeColor As Long, BackColor As Long, _  
    ShowIt As Boolean)
```

```
HandleItems Forms!frmSampleForm, _  
    ShowIt:=True, ForeColor:=255, _  
    BackColor:=0
```

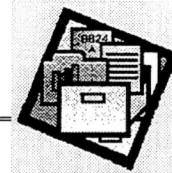
000

Optional Parameters

- ◆ Allows you to optionally pass parameters
- ◆ Must be at end of parameter list
- ◆ Once an optional parameter appears, all the rest must be optional
- ◆ All optional parameters must be Variants
- ◆ Can't use with ParamArray
- ◆ Use `IsMissing()` to know if an item has been omitted



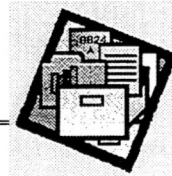
An Example:



```
' In basDeclare
Function MsgBox2(Prompt As String, _
Optional Buttons As Variant, _
Optional Title As Variant)
    If IsMissing(Buttons) Then
        Buttons = 0
    End If
    If IsMissing(Title) Then
        Title = "MsgBox Test"
    End If
    MsgBox2 = MsgBox(Prompt, Buttons, Title)
End Function
```



ParamArray



- ◆ Pass an optional number of variants to a procedure
- ◆ The same procedure then works with any number of arguments you send it
- ◆ Must be variant
- ◆ Must be last parameter for the procedure
- ◆ Can't be optional
- ◆ See MinValue() in basDeclare

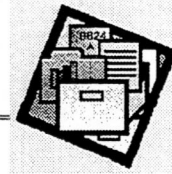


User-Defined Collections

- ◆ Create your own collections!
- ◆ Almost any data type
 - user-defined data types an exception
- ◆ Can contain other collections
- ◆ Useful when you don't know how many items before you start



User-Defined Collections



- ◆ You can specify a unique key for each item as you Add it, to later identify it
- ◆ See TestCollections in basCollections



New Data Types

- ◆ **Date**
- ◆ **Byte**
- ◆ **Boolean**
- ◆ **Object can refer to any object**
- ◆ **Variants can hold arrays**
 - **See varArrays() In basArrays**
 - **Array() is another useful function!**



Scoping Changes

- ◆ **Two scoping levels:**
 - **Private**
 - **Public**
- ◆ **Private and Public available in form/report modules**
- ◆ **Global also available in global modules, same as Public**



Calling Public Procedures

◆ cmdEnumerate_Click is public:

```
Form_frmSampleForm.cmdEnumerate_Click  
'or  
Forms!frmSampleForm.cmdEnumerate_Click
```





Using Public Variables

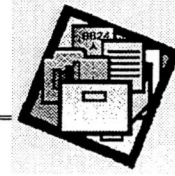
- ◆ Variables declared **Public** in form/report modules are available from outside, as if they were properties of the form/report

```
Forms!frmSampleForm.PublicVar = 12
```





Additions to MsgBox



- ◆ Add your own help button
- ◆ Use @ signs to break up text
- ◆ See TestMsgBox in basVBASamples

```
MsgBox(prompt[, buttons][, title]  
[, helpfile, context])
```



Agenda

- ◆ **VBA IDE**
- ◆ **Environment changes**
- ◆ **Language changes**
- ◆ **Working with Forms and Reports**
- ◆ **Windows API issues**

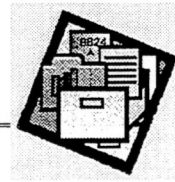


Agenda

- ◆ **VBA IDE**
- ◆ **Environment changes**
- ◆ **Language changes**
- ◆ **Working with Forms and Reports**
- ◆ **Windows API issues**



User-Defined Properties



- ◆ **Property Let/Set/Get mechanism**
- ◆ **Declare procedures to assign, set, or get property values**
- ◆ **User-defined property: BackColor**
 - BackColor property for all text boxes on the form

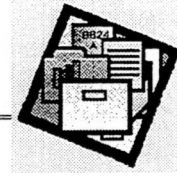


Multiple Instances

- ◆ You can create multiple instances
 - Each has own properties
 - Own record pointer
- ◆ See `basFormInstance` for example



Multiple Instance Gotchas

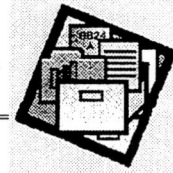


- ◆ All instances have the same name
 - So you can't reference them by name in the Forms collection
- ◆ Each has its own index in the Forms collection
- ◆ Need to store form reference somewhere
- ◆ Closing default does not close instances
- ◆ React to Close event and remove reference from the collection





Form & Report Auto-Open



- ◆ Referring to a property of a closed form or report causes it to be opened
 - It'll be hidden when it gets opened
 - Set Visible property to True to make it visible
 - Retrieving a property for a closed form also causes it to open

```
Form_frmCustomer.Caption = "Test"  
Forms!frmCustomer.Visible = True
```



Agenda

- ◆ **VBA IDE**
- ◆ **Environment changes**
- ◆ **Language changes**
- ◆ **Working with Forms and Reports**
- ◆ **Windows API issues**



Agenda

- ◆ **VBA IDE**
- ◆ **Environment changes**
- ◆ **Language changes**
- ◆ **Working with Forms and Reports**
- ◆ **Windows API issues**



Windows API Issues

- ◆ Every single Windows API call will need to be modified in your Access 95 applications
- ◆ Almost every handle is now a Long, rather than an Integer
- ◆ Need ALIAS for ANSI vs. Unicode ("A" vs. "W")
- ◆ Functions that used to return Integers now return Longs
- ◆ Common functions removed or changed





What do you do?

- ◆ **ADT and VB4 ship with declaration tool**
- ◆ **ADT and VB4 also include a useful text file of declarations**
- ◆ **MSDN CD has full help file describing all API functions**
- ◆ **Get Daniel Appleman's Win32 book!**



Questions?



